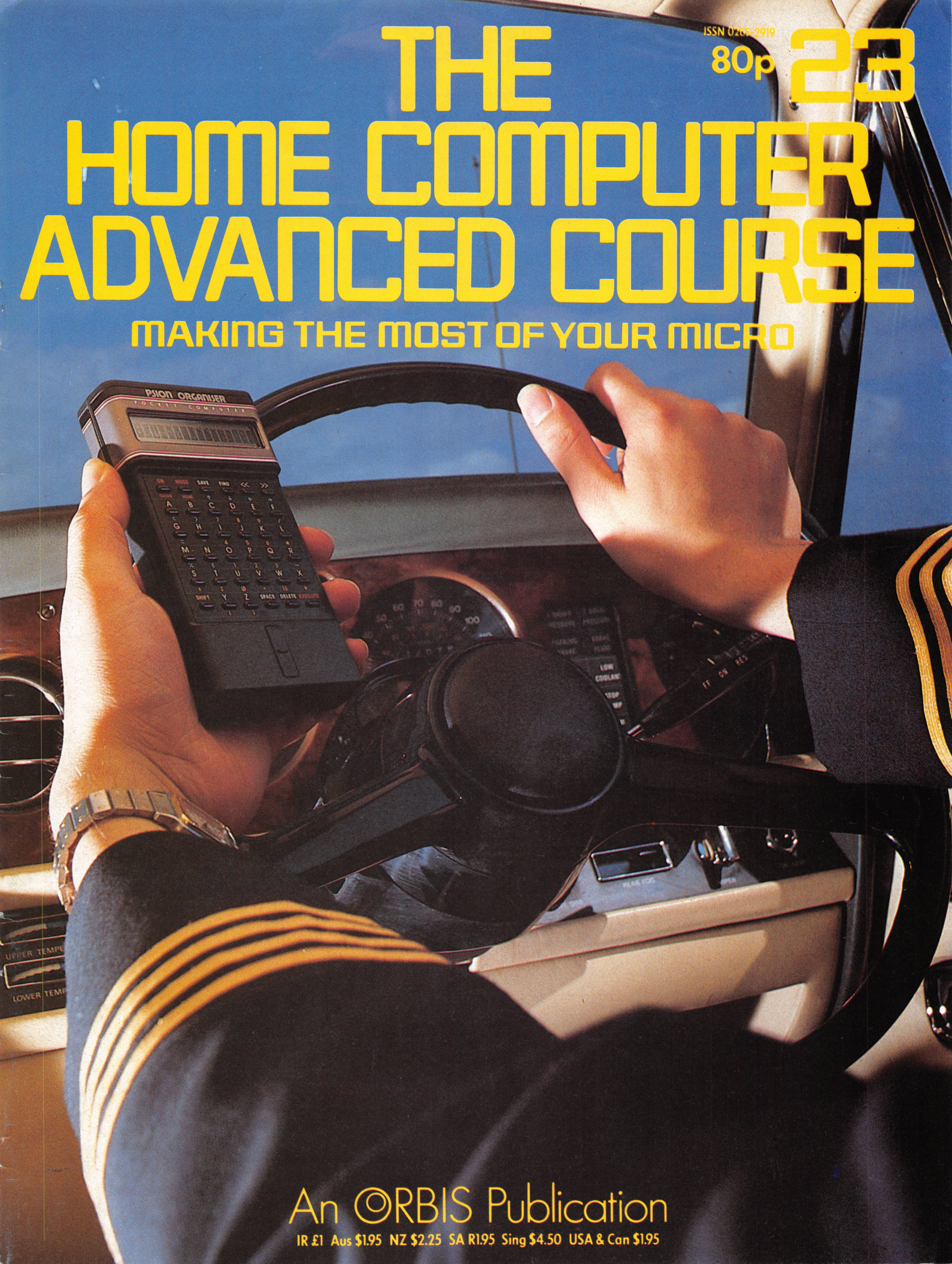
ISSN 0265-2919

80p

23

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO



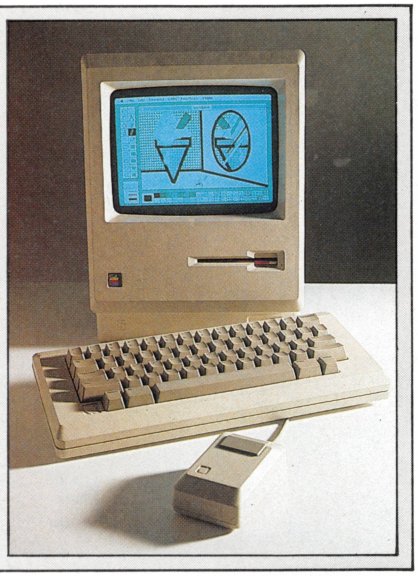
An ORBIS Publication

IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

CONTENTS

Next Week

- **We look at the Apple Macintosh, concentrating on the features that have made it a serious rival to the IBM PC.**
- **Is Big Brother watching you? We look at computer surveillance techniques, and discuss some of the implications of a highly-computerised 1984.**



QUIZ

1) The advantage of writing on a Psion Organiser is that you can erase the memory and start again. True or False?

2) Is there a facility on the Plus 1 for further interfaces to be connected on the Electron?

3) What is unusual about Alice In Videoland?

4) Does the 3D rotation program follow modular or non-modular design?

Answers To Last Week's Quiz

A1) The MicroGnome runs the micro section of 'Viewfax', the magazine section of Prestel.

A2) 0 will be added to delta-x and +1 will be added to delta-y.

A3) The custom chip that switches between BASIC ROM and screen RAM could also accommodate an extra ROM chip.

A4) Jack Kilby of Texas Instruments invented the integrated circuit.

QUIZ

COVER PHOTOGRAPHY BY MARCUS WILSON-SMITH

Editor Jim Lennox; **Managing Editor** Mike Wesley; **Art Director** David Whelan; **Technical Editor** Brian Morris; **Production Editor** Catherine Cardwell; **Art Editor** Claudia Zeff; **Chief Sub Editor** Robert Pickering; **Designer** Julian Dorr; **Art Assistant** Liz Dixon; **Editorial Assistant** Stephen Malone; **Sub Editor** Steve Mann; **Researchers** Melanie Davis, Martha Ellen Zenfell; **Contributors** Steve Colwill, Steve Malone, Geoff Nairn, Geoff Bains, Richard Pawson, Tony Harrington, Charles Arthur, Graham Storrs; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Controller** Peter Taylor-Medhurst; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 14 Rathbone Place, London W1P 1DE;

HOME COMPUTER ADVANCED COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

How to obtain your copies of HOME COMPUTER ADVANCED COURSE – Copies are obtainable by placing a regular order at your newsagent, or by taking out a subscription. Subscription rates: for six months (26 issues) £23.80; for one year (52 issues) £47.60. Send your order and remittance to Punch Subscription Services, Watling Street, Bletchley, Milton Keynes, Bucks MK2 2BW, being sure to state the number of the first issue required.

Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER ADVANCED COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER ADVANCED COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

How to obtain binders for HOME COMPUTER ADVANCED COURSE – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 5, 6 and 7. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER ADVANCED COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER ADVANCED COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

DATA COMPRESSOR

IAN McKINNELL

Psion, the company that produced the introductory software for the Sinclair Spectrum and QL, has moved into the hardware market with a pocket computer called the 'Organiser'. The machine's ability to store and retrieve large amounts of data, its wide applicability and its ease of use make it a remarkable achievement.

The Psion Organiser, priced at around £100, is the size and weight of an electric razor, and at first sight it looks like one. A rigid casing envelops the calculator-style keyboard, which has 36 small buttons. These include the letters of the alphabet (the Organiser uses upper case characters only) and several Function keys. The digits 0 to 9, some mathematical functions and punctuation symbols are all accessed using the Shift key. Above the keyboard is a 16-character liquid-crystal display, which has an adjustable contrast so that it can be read from any angle. Each character is composed of a five by eight matrix of dots.

In the back of the machine are two slots for inserting the 'datapaks' that provide the long-term storage. The Organiser comes fitted with one eight Kbyte datapak, although 16 Kbyte sizes are also available.

When the ON button at the top left of the keyboard is pressed, the display shows the time from a 24-hour clock and the date. The Organiser's four main functions are provided by the press of a single button. As the instruction manual clearly explains, there is no need to type in

Rapid File
The Psion Organiser, conceived as a pocket computer with a powerful database, provides a quick and ready filing system. With built-in calculator functions and the ability to store and retrieve vital information very quickly, the Organiser can handle some of the functions usually restricted to larger desktop or hand-held microcomputers

commands. These main functions are:

• SAVE: Data records up to 200 characters long can be entered from the keyboard into the display, edited with the two cursor controls (left and right) and the Delete key, and then SAVEd to either of the datapaks.

Contrasting Systems

For the purposes of evaluation, we have compared the Psion Organiser with a sophisticated loose leaf address/diary/notebook system

	PSION	FILOFAX
PRICE	£100	£50 (depending on your choice of pages)
SIZE	142×78×29.3mm	180×124×30mm
WEIGHT	225g	200g
DISPLAY	16 character LCD	User's handwritten/ typed entry
STORAGE	10,900 characters in 8 Kbyte rampak	Up to 200 pages
SEARCH FACILITIES	From any clue of up to 15 characters	Manual, alphabetical index
PACKAGES AVAILABLE	8 or 16 Kbyte rampak, scientific, engineering, maths, accounting and finance, restaurant listing	Diary, telephone/ address pages, cost accounting, spreadsheet, maps, notepaper and music manuscript
OTHER FACILITIES	Tells the time and date. Can be used as a calculator	Comes in smart leather wallet that can store credit and business cards

• FIND: This function retrieves data from either datapak. A search can be conducted for a string of up to 15 characters. The required string is entered into the display, and the machine matches this against the stored records. The FIND command is very fast: a typical time for searching through a database of 21 Kbytes is eight seconds. Records containing the required string are displayed in the order that they are located. A record can be scrolled in either direction using the cursor keys.

We used Psion's Restaurant Guide To London datapak, a 16 Kbyte database giving the names of 105 restaurants, each with an address, telephone number, nearest underground station, type of food, opening hours, price range indicator, and a general comment. A search for the string 'ARB' gives a match for a restaurant called 'BARBARELLAS' and also a restaurant near 'MARBLE ARCH' underground station. This function compares favourably for speed and convenience with many microcomputer database programs.

• ERASE: This deletes a retrieved record from its datapak. There is no ERASE function key on the Organiser's keyboard, and this command can be accessed only after the FIND command has been used. With the record on the display, the user presses the Mode key to obtain the ERASE facility. Pressing the Execute button will then erase the record from memory. However, this does not free any storage space for re-use, it merely renders the file unavailable to the CPU. We will discuss the nature of the datapak memory later in this article.

• CALC: The Organiser can be used as a four-function (add, subtract, multiply, divide) calculator using this command. The output can be given to seven significant figures, and scientific

KEVIN JONES

Handheld Power
The Psion keyboard contains a full alphabet, laid out in alphabetical order. This arrangement is not intuitive and may signal difficulties for people accustomed to a QWERTY keyboard. The SHIFT key provides access to a full numeric keypad, calculator functions, cursor movement and built-in database commands

notation is used for very large or small numbers. Once again, there is no key labelled CALC on the Organiser's keyboard: it is accessed by pressing the Mode key at any time other than after a FIND search has been executed.

The simplicity and ease of use of the machine is matched by its low price. The Organiser is commercially viable because low-power technology is low-cost as well. The machine can run on a single nine-volt PP3 battery for up to six months, because the circuitry uses CMOS (complementary metal oxide semiconductor) chips and the datapaks are EPROM (erasable programmable read-only memory) chips, both of which have very low power consumption levels. Until recently, the cost of these types of chip would have made the machine too expensive to have sold well.

At the heart of the Organiser is a Hitachi 6301X eight-bit processor, running at 0.93 MHz and controlled by four Kbytes of ROM. There are also two Kbytes of RAM for entered data, screen information and calculator 'working space'. EPROM datapaks slotted in the bottom of the case are the equivalent of RAM memory in other microcomputers. EPROM can be thought of as 'once-only' RAM: data written to it is stored and cannot be removed, although it can be accessed as if it were normal ROM. Thus it is 'programmable' ROM. Using the ERASE function of the Organiser on a record in the chip does *not* free its storage space in the EPROM, but marks the record to indicate to the processor that it is deleted. Thus, the datapaks eventually fill up with data until no new records can be added. The only way to erase the stored data and restore the chip to its original 'clean' state is to expose it to intense ultra-violet light. This is the function of the Psion Formatter, which costs about £40. New datapak chips cost £13 (for eight Kbyte chips) or £20 (16 Kbyte chips) and can be formatted up to 100 times.

The advantage of EPROM over RAM is that it requires no further power to retain data once it has been stored, whereas RAM is 'volatile' and must have a continuous power source. EPROM chips are also very reliable, and byte-for-byte cost one fifth as much as RAM. They are commonly used by computer manufacturers in prototype systems in place of ROM because the chips can be re-used and re-programmed in the event of software errors, whereas an incorrectly programmed ROM must be thrown away. ROM can be used once all the faults in the EPROM-held software have been found.

Communication with other computer devices is possible via an RS232 interface costing £40, which fits into one of the datapak slots and allows data to be transmitted or received at speeds of up to 9,600 baud. The protocols can be programmed using the software provided with the interface so that, for example, output to a printer can be formatted for page lengths and line width.

The Organiser can be used in a wide variety of situations in which a portable and powerful computer is needed. It could be used to note experimental results or for keeping an engagements diary, although the keyboard seriously limits typing speeds because it is set out alphabetically. The machine would be best used for storing information that requires a large amount of cross-referencing — for example, salesmen's price lists, the indexed contents of a library, or the names, addresses and telephone numbers of friends. It could also be useful in applying a computer program to experimental data on the spot, or for storing information intended for further processing elsewhere.

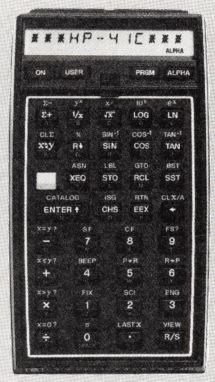
With its conveniently small size, low power consumption (there is no need for mains adaptors), storage capacities previously only possible with desk-bound machines, and effective database facilities, the Psion Organiser looks set to be a great success amongst people who wish to have a microcomputer in their pocket.

Pocket Computers

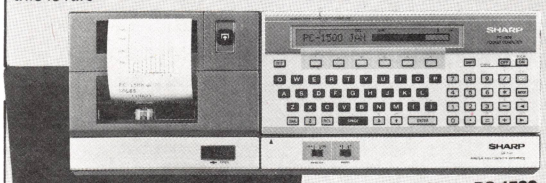
Over the past few years, several attempts have been made to market pocket computers of one sort or another. These have sold moderately well but have never achieved even a fraction of the sales of the pocket calculator.

Some pocket computers are little more than glorified calculators. These can be programmed in their own languages, but tend to appeal only to engineers and scientists who need to be able to perform complicated calculations while on the move. The next stage up is pocket computers that can be programmed in BASIC. This makes them more suitable for general use, and so programs have been produced for them that allow stockbrokers to analyse share prices, plumbers to prepare bills and airlines to sell tickets on planes.

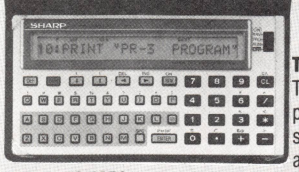
Even the top pocket computers haven't really become popular. No decent software has been produced to allow people to keep their diary or address book on them. Even if it had, the computer's memories are often too small to hold a useful amount of information. Small keyboards are difficult to use so most people would rather stick to using pen and paper for diary and address book. Computerised versions of these only offer an advantage when sophisticated search facilities are needed, and this is rare



Hewlett Packard 41C
This is a very sophisticated programmable calculator. It is available in three models, the HP-41C, HP-41CV and HP-41CX. The 41CX has ½K of memory and the other two models have 2K. Although the calculator has an alphabetic keyboard, this is only intended for programming use and so the calculator does not handle text. The price ranges from £175 to £278



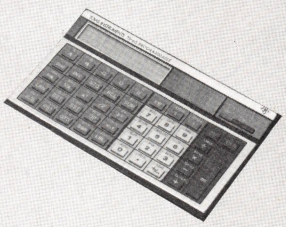
Sharp PC 1500



Sharp PC 1251

Sharp makes two pocket computers, the PC 1251 (£80) and PC 1500 (£170). The first of these is the smallest pocket computer on the market, the other is the most powerful. Both are programmed in BASIC, although the PC 1500 has extra commands. Both have 4K of memory as standard, but the PC 1500 can be expanded to 20K

Texas Instruments TI-66
This is essentially a programmable calculator and so should appeal to scientists and mathematicians. It does not handle text, so is unsuitable for more general applications. It has only half a kilobyte (K) of memory and sells for £45



Casio FX700P
This one of several pocket computers produced by Casio that can be programmed in BASIC. It has 2K of memory and a tiny alphabetic keyboard and costs £60. A version of the machine with a tiny built-in thermal printer is also produced

WINNING HANDS DOWN

Bridge, like chess, is a game of obsessive interest to enthusiasts at all levels of skill. It is ideally suited to computer adaptation, since it eliminates the need for playing partners. In this article we look at a program that teaches bridge, and programs that aim to enhance the skills of experienced players.

Commercially available bridge programs fall into two distinct categories, depending on whether they are designed specifically as teaching aids or as playing packages.

There are a number of good tutorial packages on the market. Among the best of these, in terms of design and presentation, is the Bridgemaster series. There are versions of this package available for the Spectrum, ZX81, BBC Model B, Electron and Commodore 64. This series is designed by Terence Reese, a former world champion, the bridge correspondent of the *Observer* and the London *Standard*, and the author of a number of books on bridge.

The principle behind all bridge tutor programs is that the beginner works through a series of pre-arranged hands, written into the program. This is the essential difference between a tutor and a playing package, which generates random hands. Because the tutor 'knows' what each of the hands contains, the program is able to guide the player step by step through the many rules and conventions that make bridge such an enjoyable and intellectually demanding game.

The Bridgemaster series provides an excellent example of what can be achieved by a tutorial package. There are two programs in the course, which are aimed at the complete beginner and the average player respectively. The first is called the Complete Learning Package For The Beginner At Bridge, the second is Expert Bridge. The beginner's package consists of two program cassettes, two commentary tapes, a slim instruction booklet and Reese's *Begin Bridge*, a paperback book published by Penguin. This last is not meant to be used at the same time as programs on the screen. The commentary tapes provide all the backup the beginner needs.

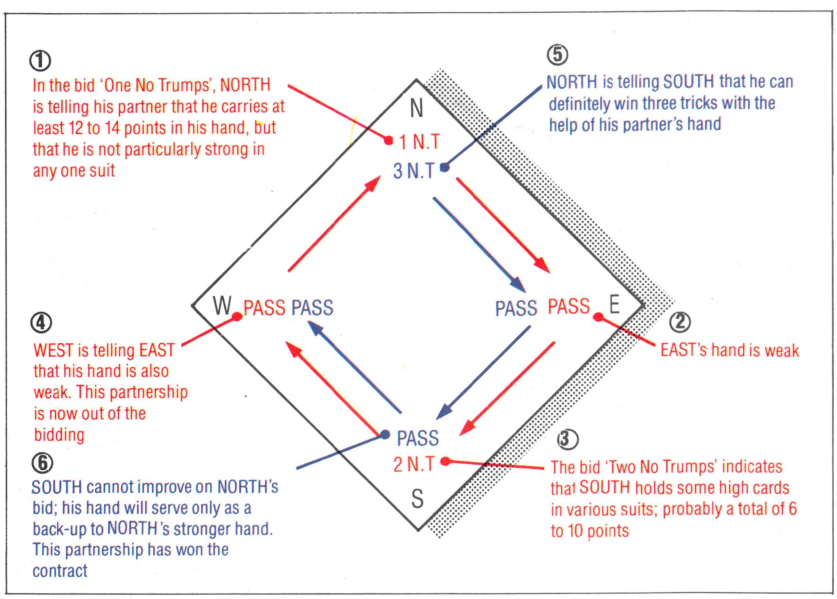
Bridge tutor programs that do not have a tape-based commentary have to provide either a very comprehensive manual or extensive narrative commentary on the screen. Alternatively, they can assume at least a basic knowledge of bridge and restrict themselves to improving your play.

For those who do not know the game, the pack of cards is first dealt out to the four players who pair into teams of two. In computer bridge, the standard convention of naming the four players (North, South, East and West) is generally adopted. Bidding precedes play, in which the players, without showing their hands to each other, must declare the strength of their hand of cards. The strength of a hand is determined by the number of high cards or the number of cards in any one suit or both. In our diagram, we show an example of bidding. The winning bid will determine the contract. This selects what suit is to be trumps as well as the number of tricks that the winning side must try to make. After bidding, the hand is played. In the play, the player that has won the bidding becomes the 'declarer' and his partner the 'dummy'. The dummy's cards are displayed for all to see. The declarer will win or lose points according to whether he takes the number of tricks specified in the contract.

All bridge tutor programs represent bidding, whether or not they allow the player to win it properly. In Bridgemaster, the player is given the opportunity of making his own bid, having seen his cards displayed on the screen. The computer will then declare the bid that is to be made. This may differ from the player's choice of bid. Reese's program, like many tutor programs, is rigged so that the computer will accept only the bid or the play that the program's designer wants you to make. Reese admits that this leads, on occasion, to a perfectly sensible play by the beginner being rejected in favour of the predetermined route that has been mapped out.

Besides displaying your hand and the bidding, the program has to deal with the play of the hand. The most common method of display is a square in

Successful Bid
The values of the cards in the bidding are: Ace–4; King–3; Queen–2; Jack–1. The type in red refers to the players' first bid, and the type in blue shows their second bid



the centre of the screen with the four sides labelled N, E, S, W. North and South's hands are then fully displayed. Usually the four suits are shown: spades, hearts, diamonds and clubs, ranked in that order, with the card values displayed alongside.

The player is assumed to be South, and either North or South will be the dummy. In either event, South always plays the cards for both. In other words, South always plays the hand, irrespective of whether North or South won the bidding. This is, of course, a departure from actual bridge play, in which you can spend an entire evening without once winning the contract.

In a tutorial program this departure does not matter, except in the important area of defensive play. These skills are needed when you want to stop your opponents from fulfilling the contract once they have won the bidding. In a playing program, the implications are more serious, and we do not know of a playing program that will play out a sequence in which East and West win the bidding. If you 'duck out' of the bidding in, for example CP Software's excellent playing program for the 48 Kbyte Spectrum — in other words if you deliberately allow East or West to be declarer — then the program prints a message on the screen telling you that it is not designed to play under such circumstances, and instructs you to press the R key to generate a new hand.

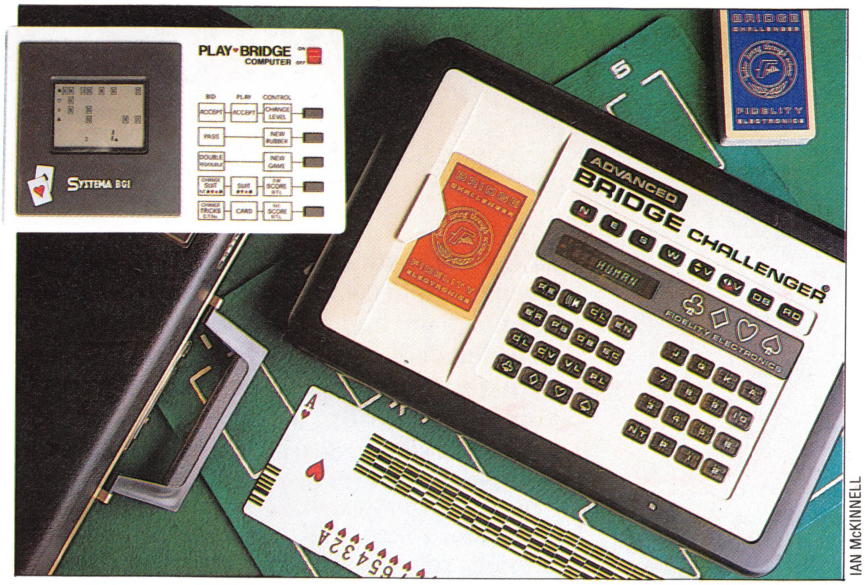
When the bidding has been completed, the play of the hand is displayed. Depending on who wins the trick, either the computer (playing for E and W) or the player (for S or N) selects a card. E and W's cards appear on the screen one by one, as they would on the table during normal play.

There are a number of useful features that computer-based programs offer. Bridgemaster, for example, lets the player choose from four options before each deal depending on the level of competence. P plays the hands out with the human player selecting plays for N and S (under Reese's guidance); A selects 'autoplay', where you watch the hand played out automatically; H displays all four hands on the screen, while D selects a different deal. This is a standard set of options with most bridge programs.

The autoplay feature is particularly useful in combination with Reese's commentary, since the player does not have to be distracted by the task of selecting cards. All deals can be replayed over and over again. Playing programs have these features as well, but here they act not as repeated drills of teaching points, but as post mortems on the bidding and the play.

We have looked at some of the principles of tutorial packages, but there are also a number of programs on the market for the more advanced player who wishes to improve his game.

The Expert Bridge package in the Bridgemaster series that we have discussed previously is a well presented advanced instruction course that looks at the complexities of such things as high point hands, slam bidding, squeeze and blocking plays.



IAN McKINNELL

Dedicated Challenge
The two machines pictured are bridge-playing computers. The first (right) is a table-top game called Advanced Bridge Challenger, by Fidelity Electronics. Bridge Challenger is recommended only for experienced bridge players as it requires a great deal of practice and concentration to master. A major drawback of the Bridge Challenger is its tiny 8-character screen. It sells for £169.95. The second machine (left) is the Systema Play-Bridge computer. This operates on batteries and costs £29.95. Play-Bridge takes only a few minutes to understand; it doesn't have many advanced features, but it is great fun to play

Another package worth looking at for the more experienced player is the playing program from Alligata Software. Versions are available for the Oric, BBC Model B and Electron and the Commodore 64. A fair amount of bridge knowledge is assumed by the program and there is no manual provided with this package, so players without good memories had better be prepared to take notes.

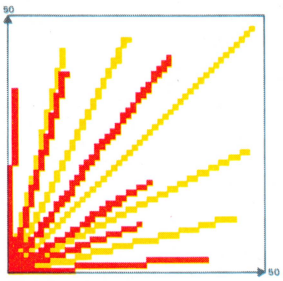
CP Software's program for the Spectrum is a very enjoyable program to play. It will not play the hand as declarer, so there is no way to practise your defensive play. However, since most amateurs and social players, who make up 98 per cent of the world's bridge playing fraternity, want to play every hand anyway, this is not a great disadvantage.

Bridge programs, both of the tutorial and playing kind, are an ideal way of learning the game and improving on your weak points. Bridge is a game that demands so many different skills: a good memory, analytical skills, the ability to work in a partnership, the ability to play with caution and to know when to take a risk, and the ability to remain inscrutable and to keep a cool head. The games that we have discussed will not perfectly simulate a bridge game between four players, but they will certainly give you worthwhile and stimulating practice.

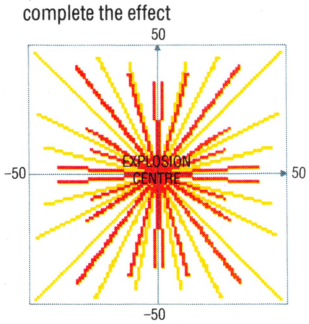
IAN McKINNELL

Pick A Package
There are a number of bridge packages available, ranging from teaching programs to challenging games that test the skills of advanced players. Shown here (left to right) are Bridgemaster, available on cassette for the Spectrum, ZX81, BBC Model B, Electron, and Commodore 64 at £24.95 each; Bridge from Alligata Software, available on cassette for the Oric, BBC Model B, Electron and Commodore 64 at £8.95; and Bridge Player from CP Software, available on cassette for the 48 Kbyte Spectrum at £9.95

BLOW-UP



Explosive Play
The explosion graphics are created by generating lines of random length within a range of −50 to 50 units from the centre. The explosion is built up in quadrants and mirrored to complete the effect



In the last section of our project to build up our Minefield game for the BBC Micros and the Electron, we looked at how the program detected collisions with the mines. We now take the program a step further towards completion with a discussion of the procedures we will use to create the visual and sound effects of explosions.

The BBC Micro and the Electron support 16 possible colour variations that we can use in creating an explosion effect. There are actually only eight different colours, the other eight variations being flashing effects between one colour and another. Normally, the flashing combinations of colours alternate every half a second, but the flash rate can be changed by two FX commands. *FX9 sets the amount of time that the first colour of the flashing pair is displayed. The time is measured in units of fiftieths of a second. Thus, *FX9,20 alters the display time of the first colour to 2/5ths of a second. The other FX command, *FX10, can be used to alter the display time of the second colour in the same way. By altering one or both display times, interesting flickering effects can be produced.

The visual explosion effect can be produced by drawing a series of short lines in various colours out from a central point. To enhance the effect, the colour of the mines and score lettering can also be rapidly changed. As these were all originally PRINTed in logical colour 2 (which we set to green) we can use the following command to reassign logical colour 2 to a colour randomly selected from the 16 possible combinations:

```
VDU19,2,RND(15),0,0,0
```

RND(15) selects a whole number from 1 to 15. This means that colour 0 is never selected, but as colour 0 is black, the background colour, this does not matter.

The colour used to plot the explosion lines can also be randomly selected using GCOL 0,RND(3). We then use a loop to draw a series of lines from the centre of the explosion in a series of randomly selected colours. Up to now we have looked at the high resolution commands MOVE and DRAW. DRAW(X,Y) always draws a line to the point with 'absolute' co-ordinates (X,Y). However, there is another family of high resolution drawing commands that allow us to, among other things, specify points 'relative to' one another. The PLOT K,X,Y command can be used to draw lines between relative or absolute points depending on the value of K. The following table shows some of the variations possible using PLOT:

K=0	Move relative to last point
K=1	Draw line to relative position in foreground colour
K=2	Draw line to relative position in logical inverse colour
K=3	Draw line to relative position in background colour
K=4	Move to absolute position
K=5	Draw line to absolute position in foreground colour
K=6	Draw line to absolute position in logical inverse colour
K=7	Draw line to absolute position in background colour

PLOT4 and PLOT5 are entirely equivalent to the MOVE and DRAW commands. For our explosion effect we shall use PLOT1 to draw lines relative to the centre of the explosion. Before we can draw any lines, however, we need to know where the centre of the explosion is.

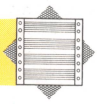
In the last section of the course, when we called a dummy version of PROCexplode, we passed the values of xgraph and ygraph to x-explode and y-explode. These played no part in the dummy procedure but will now be used to specify the centre of the explosion. If this procedure is called from line 3390 of PROCmove (see page 435), then xgraph and ygraph will be the graphics co-ordinates of the centre of the character cell containing the mine. Thus, MOVEx-explode,y-explode will move the graphics cursor to the centre of the place we wish the explosion to be. The reason why we pass the co-ordinates to another pair of variables for use within the explosion procedure, instead of simply using xgraph and ygraph, is that this procedure will also be called from another part of the program where the co-ordinates of the explosion centre will be specified by a different pair of variables. By passing parameters into the explosion procedure, we have made it more flexible.

Once we have MOVEd to the centre of the explosion, we shall draw a line in a random direction to, say, a maximum of 50 units:

```
PLOT 1,RND(50),RND(50)
```

will not quite do the job, as it will not specify negative co-ordinates. (A negative x co-ordinate produces a line drawn to the left, while a negative y co-ordinate gives a line drawn down from the explosion centre). Repeated use of this command will fill only one quarter of the space around the centre (where both co-ordinates are positive).

So that we can introduce negative values (and hence draw lines in all directions around the explosion centre), we must replace RND(50) with RND(100)−50. The maximum value of RND(100) is, of course, 100, so the most that RND(100)−50 can be is 100−50 = 50. The minimum value of RND(100), on the other hand, is one. Hence,



RND(100) can be as small as 1−50 = −49.

By repeatedly MOVEing to the centre of the explosion and then drawing a relative line up to 50 units in any direction and colour, we can produce a visually exciting effect.

Now let's consider how we can produce sound effects to accompany the explosive graphics on screen. It is possible to create quite sophisticated sounds using BBC BASIC's SOUND and ENVELOPE commands. SOUND can generate either pitched notes (so you can play a tune) or 'noise' for special sound effects. The ENVELOPE command is used in conjunction with SOUND to 'shape' the sound you hear, and thus simulate musical instruments.

SOUND has four numbers (or parameters) associated with it, which control the characteristics of the tone produced. These are as follows:

SOUND C,A,P,D

- C is the channel number, which must be a value between 0 and 3. Channels 1 to 3 produce pitched notes but channel 0 is for special sound effects. This is the channel that we shall use here.
- A is the amplitude, or, in plain terms, the volume of sound produced. Maximum volume is set when A is −15 and ranges to silence when A is zero. Positive values of A are used to select the shape of the sound and are used together with the ENVELOPE command.
- P is usually the pitch of the note — in other words how high or low the sound is.
- D is the duration or length of time that the tone sounds for. Values of D from 0 to 254 cause the note to hold for a time measured in units of 1/20th of a second. Hence, to hold a note for one second the value of D must be set to twenty. If D has the value −1 then the sound will continue until you take action to stop it.

The explosion sound effect clearly needs to have maximum volume, and thus our command will have the maximum amplitude, signified by a value of −15. The pitch value is slightly more complex. When used with channel 0, P takes values from 0 to 7 according to these rules:

P=0	High frequency pulse
P=1	Middle frequency pulse
P=2	Low frequency pulse
P=3	Pulse, frequency set by channel 1
P=4	High frequency noise
P=5	Middle frequency noise
P=6	Low frequency noise
P=7	Noise, frequency set by channel 1

The group of four pulse waveforms produce 'buzzer'-type sounds, whereas the noise waveforms produce what sounds like a television set before it has been correctly tuned. The SOUND command we will use in our program is:

SOUND 0,−15,4,40

which produces a high pitched hiss, lasting two seconds (40/20) with the volume set to maximum. Here is the complete listing for the procedure:

```
3550DEF PROCexplode(x_explode,y_explode)
3560REM ** SOUND EFFECT **
3570SOUND 0,-15,6,50
3580REM ** SET FLASH RATE **
3590*FX9,20
3600*FX10,50
3610FOR I=1 TO 100
3620MOVE x_explode,y_explode
3630VDU19,2,RND(15),0,0,0
3640GCOL 0,RND(3)
3650PLOT 1,RND(100)-50,RND(100)-50
3660NEXT I
3670PROCreset
3680ENDPROC
```

THE 'RESET' PROCEDURE

After an explosion there are several tasks to do:

- We must reduce the number of lives remaining and check to see if all the lives have been used.
- We must clear up the mess made on the screen by the explosion.
- We must reposition the detector and assistant in their starting position.

The first of these is easily accomplished by use of a count that is incremented each time the procedure is called. If the count exceeds four after incrementation then a special variable, end flag, is set to one to signal the fact that the game is over.

Having tested for the end of game, we have to clear up the explosion. Several methods could be employed here, such as PRINTing blank spaces over the area of the explosion. However, the method we use is to clear the screen, relay the mines and reprint the time and scores information. To make it slightly fairer, we should relay only the same number of mines as were left before the explosion. This can be easily calculated from the score. As each mine is worth 150 points and there were 50 originally, we can calculate the number of mines left and pass it to the 'lay mines' procedure.

The mine detector and assistant are repositioned by re-initialising their co-ordinates (by calling the initialise variables procedure), and then calling the position chars procedure. The full Reset listing is shown below. Add this and the explode procedure to your program.

```
3880DEF PROCreset
3890count=count+1
3900IF count>4 THEN end_flag=1:ENDPROC
3910CLS
3920VDU19,2,2,0,0,0
3930COLOUR 2
3940PROCinitialise_variables
3950mines_left=50-score/150
3960PROClay_mines(mines_left)
3970PROCdraw_border
3980PRINTTAB(2,27);"Time"
3990PRINTTAB(2,28)"Score"
4000PRINTTAB(11,28)score$
4010PRINTTAB(2,29)"Hi score"
4020PRINTTAB(11,29)hi_score$
4030remaining_men$=LEFT$(men$,4-count)
4040COLOUR 1
4050PRINTTAB(2,30);remaining_men$;"    "
4060COLOUR 2
4070PROCposition_chars
4080ENDPROC
```

Also, the following modification is needed on the temporary calling program (see page 405):

60 UNTIL TIME>12099 OR end_flag=1

KEVIN JONES

D

DIGITAL PLOTTERS

Plotters used by home computer owners are, almost invariably, digital in design. They are connected to the micro via the printer interface and are used to interpret bytes of data in such a way as to produce (x,y) co-ordinates, and in some cases different colours, as printed output. Earlier plotting devices were analogue in nature, and these were widely used in scientific applications. For example, an analogue plotter would be used to show how the temperature of a furnace varied over a given period of time.

DIGITISE

To *digitise* is to convert analogue information into its digital equivalent. One application is the digitising of speech: the output of a microphone is fed into an analogue-to-digital converter, and the resulting bytes of data are stored on disk. By reversing the process, the computer can reproduce speech in a far more intelligible form than a speech synthesiser can. However, in such applications, the term 'sampling' is more common than 'digitising', because the signal is being sampled and measured at regular intervals. It can be proven mathematically that to reproduce a signal accurately it must be sampled at least twice as often as the highest frequency in the signal. Thus, if we assume that speech entails frequencies from 300Hz (hertz—cycles per second) to 3,400Hz, the signal must be sampled 6,800 times per second.

DIGITISER

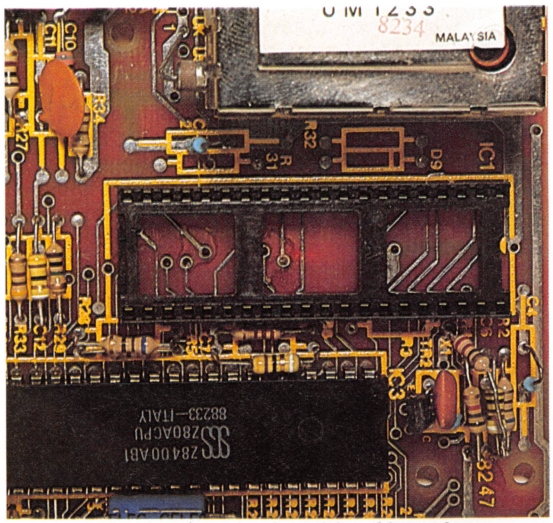
Many applications for computers, notably in the fields of engineering, surveying and design, require that a visual image on paper be transferred to the computer. The device needed for this is a *digitiser.* The most advanced units are essentially television cameras coupled to very fast A-D converters (see page 28), but, more commonly, the image has to be traced over by hand, using a special stylus.

The means by which the digitiser detects the position of the stylus, and thereby creates the digital values to represent the image, varies considerably. Some units employ a mesh of wires embedded in the base plate of the unit. In this system, the stylus traced over the image picks up the magnetic field from the wires. Ultrasonic systems exist, too, in which two sound detectors measure the distance from the pen, and calculate its absolute position.

DIL

To reduce manufacturing costs, most integrated circuits (chips) are soldered directly onto the printed circuit board. They are inserted into the holes on the board using automatic insertion machines, and then the whole assembly is dipped into a bath of molten solder.

On older microcomputers, the chips weren't soldered to the board but were inserted into *DIL* (Dual In-Line) sockets that were soldered onto the board. The DIL name simply comes from the two lines of pins that characterise integrated circuits.



IAN McKINNELL

The photograph above is a printed circuit board from a home computer showing a DIL socket

Despite the increased manufacturing costs, the advantage of using DIL sockets is that the replacement of faulty chips doesn't entail any soldering. However, RAM chips are now far more reliable than they were 10 years ago — and, faced with a fault, many manufacturers now find it cheaper to replace the whole board than to de-solder an individual chip.

Some home computers feature empty DIL sockets on their printed circuit boards. These are particularly useful, as they allow additional ROMs (for a disk operating system or graphics utility language, for example) to be fitted by the user.

DIMENSION

The number of *dimensions* in a system is the number of parameters that must be supplied to specify a unique point in that system. We are used to three-dimensional spatial systems, where co-ordinates on three axes define a point in space.

In computing, the concept of dimensions becomes relevant when we deal with array variables. The array A(5,6) is two-dimensional, because two quantities have to be given to access any particular variable element. Note that the size of the array is something quite different: in this case it is 30 (5 × 6) elements. Some BASICs will allow only one or two-dimensional arrays.

DIRECT ACCESS

This is really a better term for random access — particularly when referring to disk files. *Direct access* means that any item within a file can be accessed immediately, without the need to work through a lot of other items — as is the case with a sequential access file. If a file is composed of records — as in a database, for example — this means that the disk operating system must keep a constantly updated set of pointers to indicate where each record starts within a file. When a particular record is requested from disk, the DOS looks up its position from this table, and then moves the head directly to the appropriate track and sector.

PERFORMANCE BOOSTER

The Acorn Electron was originally developed to provide some of the features offered by the BBC Micro at a lower price. However, the machine lacks many facilities required by the home user. The new Electron Plus 1 interface, priced at £60, can transform a standard Electron machine into a low-cost alternative to the BBC Micro.

The Electron was conceived as a scaled-down version of the BBC Micro. Its designers retained the excellent BBC BASIC and operating system, but dispensed with most of the interfaces that make the BBC Micro so versatile. In fact, the Electron's only connections are a cassette port, two monitor sockets (RGB or composite video), a modulator output for connection to a television, and a power socket.

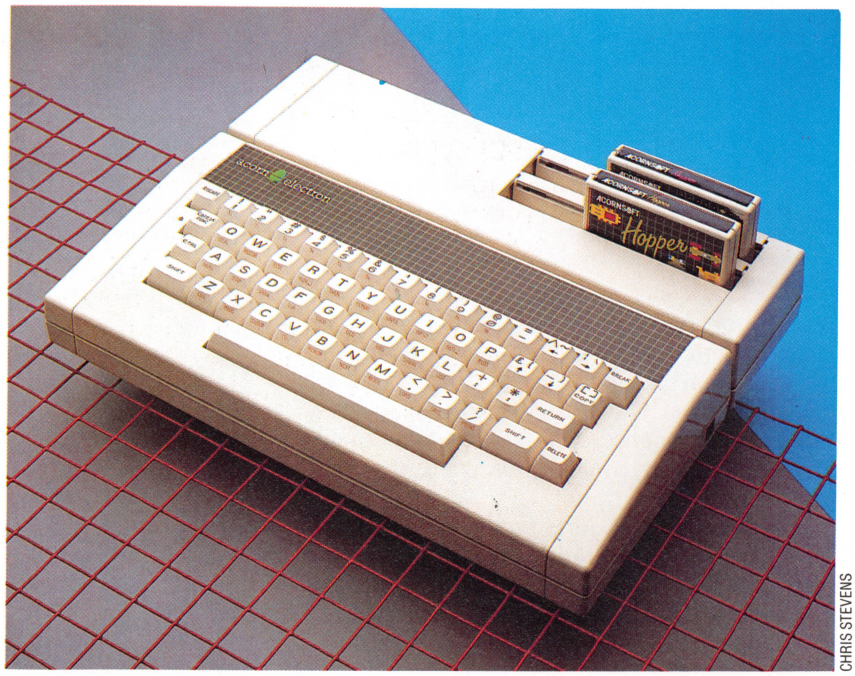
The Plus 1 interface adds to the standard machine a parallel printer port, a joystick socket and two sockets for plug-in ROM cartridges. Acorn hopes that the extra facilities will increase sales potential. The cartridge slots and joystick port allow the Electron to be used for arcade-style games, while the printer port is a useful feature for the user who needs to be able to print out program listings, or who wishes to use word processing software.

Installation of the Plus 1 expansion unit is very easy: it simply pushes onto the edge connector that protrudes from the rear of the Electron. Two bolts are then pushed through the Plus 1 casing and attached to the threaded sockets in the Electron casing. All power and data are transferred via the edge connector. The extra software needed to 'drive' the new interface is held in a ROM chip inside the unit.

The printer port is a standard Centronics parallel type. To enable the printer, the user must type the command VDU2, and to disable it, VDU3 must be entered. These commands are identical to those used on the BBC Micro. Similarly, VDU1 is used to send a character to the printer; again this is standard BBC BASIC.

Joysticks are plugged into the analogue port. This is a 15-pin D-type socket, and pin connections are identical to those on the BBC Micro's analogue port, allowing any BBC-compatible joystick to be used. The Plus 1 can measure up to four analogue voltages, either from four games paddles or from two joysticks (each joystick produces two voltages: one for up/down movement, the other for left and right).

The process of converting analogue voltages into digital signals that can be handled by the computer is performed by an analogue-to-digital converter chip — in this case, an ADC0844. This is a less complex chip than the one used in the BBC Micro, which means that the Electron does not follow the movement of a joystick as accurately as a BBC machine does. In games programs, where the joystick is used simply as a four-way switch, this is not a problem. If the joystick is used with a graphics program — to draw 'freehand' on the screen, for example — movement is noticeably jerky on the Electron and the resulting picture is less detailed.



CHRIS STEVENS

Room For Expansion
The Electron was originally conceived as a cut-down version of the BBC Micro. It lacks almost all of the BBC's many interfaces, yet still offers the same excellent graphics and BASIC at half the price. Now Acorn has produced a unit known as the Plus 1, which gives the Electron the most important interfaces for £60

When the Electron Plus 1 is first switched on, the cartridge in the front ROM socket starts running automatically — to stop it, Escape must be pressed. The ROM filing system works very much like the cassette version (it's much faster, though), so commands like *CAT, LOAD and CHAIN are understood. To use cassette software with the new unit, you must enter the command *TAPE or remove the cartridge, remembering to switch the power off first. In addition to games programs, other languages will also be produced in cartridge form. Unlike the games ROMs, these language ROMs are 'switched in' to replace the usual BASIC ROM.

The problem with most computer add-ons is that they often have unwanted side-effects. When the Plus 1 unit is connected to the Electron, loading errors may result when cassette programs are used. This is especially true for programs that contain data files — we found that two of the programs on the Electron's introductory cassette

Software Six
The cartridges for the Electron have an advantage over tape since they take only a couple of seconds to load compared to the several minutes taken by tape. Only six titles are available in cartridge format. These consist of four games and an educational program, costing £12.80 each, and the LISP language at £40

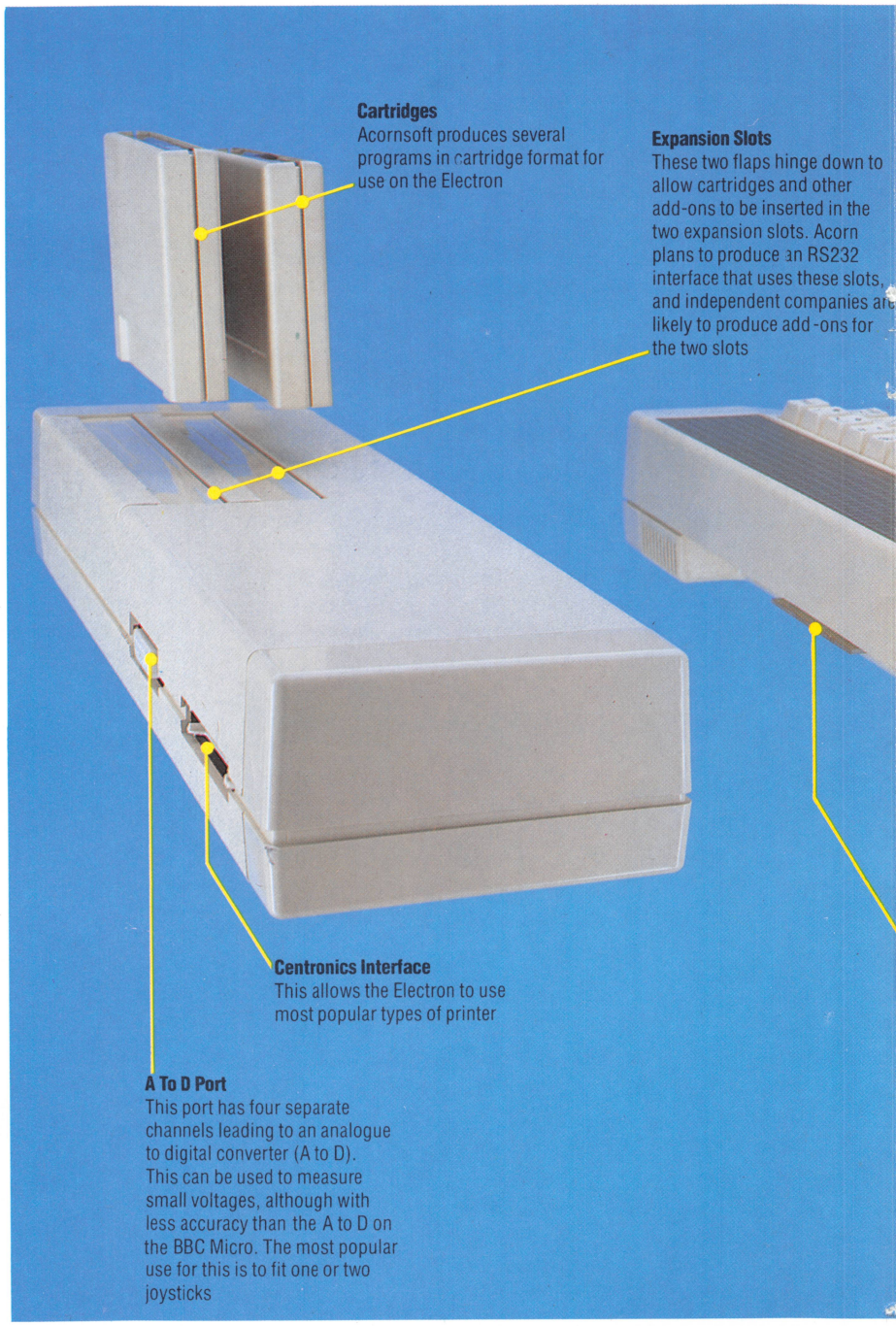
would not load on the expanded Electron. By using a series of operating system commands, however, you can 'fool' the Electron into thinking that the Plus 1 is not connected and so restore normal performance. This is barely mentioned in the manual and could prove confusing to the novice user.

On any Acorn machine, typing *HELP gives a list of the various ROMs inside the computer. Typing this on the Electron with the Plus 1 unit fitted tells you that the main operating system ROM is OS 1.00, and also lists Expansion 1.00 ADC/Printer/RS423. This is the operating system ROM inside the Plus 1 expansion unit, but the interesting thing about this is the last piece of information — RS423. This is a reference to a standard serial interface, yet neither the Electron nor the Plus 1 supports such a facility. Acorn is, in fact, planning to introduce a serial interface at a later date.

The major difference in the two BASICS is that the Electron lacks Mode 7, the teletext-compatible graphics mode. This mode is used for titles and instruction pages in BBC Micro programs as it is economical in terms of memory. On the Electron, the various viewdata attributes do not turn on the Mode 7 colours and flashing letters, but are instead displayed on the screen as gibberish. Most games programs use a different mode for the actual play, so there's no problem once the title page and instructions have been bypassed.

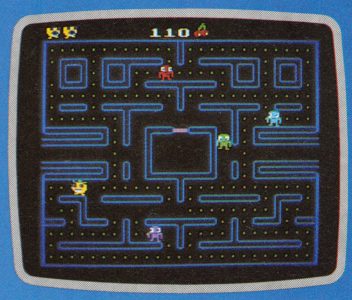
The other major difference between the two machines is in the SOUND and ENVELOPE commands. The Electron has a solitary sound channel, instead of the four offered by the BBC machine. Similarly, the Electron ENVELOPE command affects the pitch only, and not the volume. The commands are compatible, though, so a program that uses them will work on both machines — although the sound will be noticeably different on the Electron.

The Electron keyboard has a better 'feel' than that of the BBC Micro, but it has fewer keys. This means that on the Electron, as on the Sinclair Spectrum, most keys have three or four different functions. For example, the key marked 'L' will produce either 'l' or 'L', depending on whether the



Shift key has been pressed; but if used in conjunction with the Function key the 'L' key will generate the BASIC command LIST. Pressing the Control key at the same time as 'L' will clear the screen. It would perhaps have been better if Acorn had retained the 10 red Function keys provided on the BBC Micro, as these can be programmed to perform various useful tasks. The Electron does have Function keys, but these are merely the numeric keys pressed in conjunction with the Function key.

The one area in which the two machines differ dramatically is in the range of interfaces provided. The BBC Micro has more interfaces than any other home computer, which means that a host of peripherals — from printers, modems and disk drives to second processors and robot arms — may



Snappy Play
Acornsoft's Snapper is quite a good version of the classic PacMan game. It costs £12.80 in cartridge format and can be controlled with keys or with a joystick. The game is also sold in tape format at a lower price

Expansion Connector
The Plus 1 links to the Electron via this edge connector. This is the only provision for expansion on the Electron

easily be attached. All of these require additional chips and sockets, which inevitably add to production costs. By omitting these interfaces, not only are the extra chips unnecessary but power requirements are less and the computer is smaller and cheaper. That was the philosophy behind the original Electron. The fact that Acorn has now introduced the Plus 1 suggests that possibly this cost-cutting went too far.

The Electron Plus 1 is less versatile than the BBC Micro and offers fewer expansion possibilities. But many users do not require such facilities and at a price considerably less than the BBC machine, the Electron Plus 1, with its excellent graphics and sound facilities, structured BASIC and a growing range of software, looks good value for money.

BBC MICRO MODEL B

PRICE
£399

DIMENSIONS
75×340×410mm

CPU
6502, 1.8MHz

MEMORY
32K RAM, 32K ROM

SCREEN
8 display modes. Highest resolution: text – 80×32 characters; graphics – 640 × 256 pixels. Up to eight colours, which can be steady or flashing. Teletext display mode. User definable characters.

INTERFACES
UHF for television, RGB and composite video for monitors; cassette port; RS423 and Centronics printer interfaces; analogue port (for joysticks etc.); ROM sockets (for software); Tube (for second processors); 1MHz bus; Disk interface (optional); Econet networking interface (optional); user port; auxiliary power output (to power disk drives etc.)

LANGUAGES AVAILABLE
BBC BASIC (included), 6502 Assembler (included), LISP, FORTH, BCPL, PASCAL

KEYBOARD
72 typewriter style keys. This includes 10 programmable function keys.

DOCUMENTATION
The BBC manual is an excellent guide for experienced programmers, but offers little help for novices.

STRENGTHS
One of the best versions of BASIC available, a huge range of interfaces, good graphics, sound and keyboard.

WEAKNESSES
Not easy for the beginner to get started on the BBC

ACORN ELECTRON PLUS 1

PRICE
£259 (Electron £199, Plus 1 Interface £60)

DIMENSIONS
65×260×340mm (with Plus 1 fitted)

CPU
6502, 1.8MHz

MEMORY
32K RAM, 32K ROM

SCREEN
7 display modes. Highest resolution: text — 80×32 characters; graphics — 640×256 pixels. Up to eight colours and eight flashing. User definable characters

INTERFACES
UHF output for television; RGB and composite video for monitors; cassette port; parallel printer port; analogue port (for joysticks etc.); two sockets (for ROM cartridge software etc.)

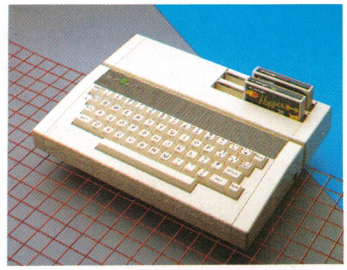
LANGUAGES AVAILABLE
BASIC, 6502 Assembler (supplied); FORTH, LISP (also on ROM cartridge), S-PASCAL

KEYBOARD
56 typewriter-style keys. Function key permits single key entry of BASIC commands. 10 programmable keys

DOCUMENTATION
The User Guide is well laid out and easy to read. It has a very thorough coverage of Electron BASIC and (most unusually) 6502 Assembly language

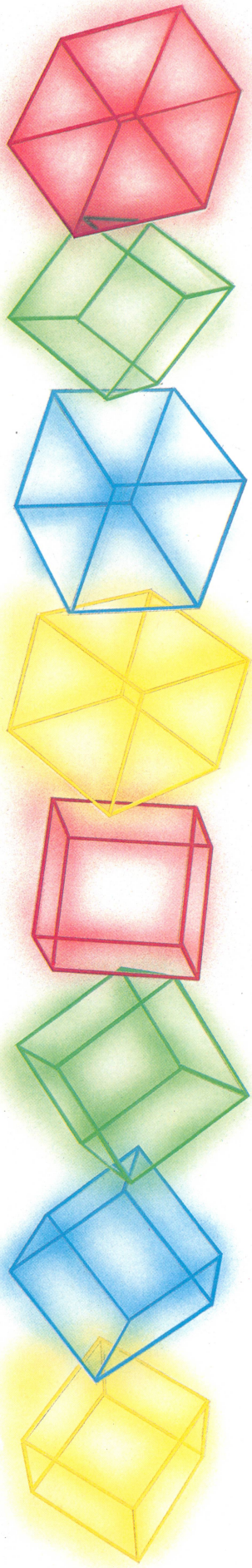
STRENGTHS
The graphics are good, and it produces a clear picture. High-quality keyboard. A good version of BASIC.

WEAKNESSES
The multi-function keys can be confusing. There is only one sound channel. Little available memory. No teletext mode. Lack of communications port



CHRIS STEVENS

CUBIST REVOLUTION



Our programming projects series continues with a look at a program that allows us to draw three-dimensional shapes and rotate them. The geometrical principles used in the program to plot the points on the outlines provide a basis on which simple animated graphics can be developed.

The program we provide uses some basic geometric principles to create perspective projections of objects that can be viewed from any angle or distance. All the data for the objects are stored in DATA statements in the program. These comprise three-dimensional co-ordinates for each end point of a line in the object. For each point there is also a figure to indicate whether it is at the beginning of a line or at its end, which tells us whether the line is to be drawn from, or to, that point.

To turn these three-dimensional co-ordinates into two-dimensional values representing points on the screen is a matter of simple, but lengthy, mathematics. The x and y co-ordinates of each point (in the plane of the television screen) are divided by a factor representing the distance of the object from us. In addition, the resulting co-ordinate is scaled by a factor suitable for the co-ordinate system of the micro. By changing the distance factor, the object can be made to shrink or grow as it approaches or recedes from the viewer.

A third constant can be used to change the effect of the perspective projection. By increasing the value of this constant the perspective view of the object is exaggerated, as though it were being viewed with a wide angle lens. Decreasing this constant gives the effect of flattening the view, as though we were looking through a telephoto lens.

In addition to creating a perspective view of the object, the program also allows it to be rotated so that it can be viewed from any angle. This is done by simple trigonometry. The axes are rotated through the desired angle so that when the perspective projection is made the image on the screen appears to have been rotated. This can be done either as a rotation around the y axis (the viewpoint appears to move around the object) or around the x axis (the viewpoint rises above or below the object).

Although our program can create all of these effects, it is remarkably simple in operation. Control of the viewpoint is through number keys one to eight. These will give, respectively: a movement of the viewpoint to the left, the right, up, down, towards the object or away from it; an increase in the perspective effect (fisheye); or a decrease in the perspective effect (telephoto).

The current three-dimensional co-ordinates are stored as three arrays: X, Y, and Z. The changes to these arrays and the changes to the constants used in the perspective transformation are made in a series of subroutines. Each time a keypress is detected, the image on the screen is erased by drawing it in the background colour, the desired change is made with a call to the right subroutine, and the new image is drawn again.

The perspective transformation is made in the subroutine that draws the image. This goes through each set of three-dimensional co-ordinates, translates them to two-dimensional co-ordinates and plots them on the screen (either moving to the points or drawing lines according to the fourth piece of data for each point).

Creating your own object data for this program is a fairly lengthy business, but very straightforward. The data is stored in statements at the end of the program, with four values for each point in the object. The total number of points is set in the first line of the program. This should be changed to suit your data. The data we give creates a cube with a diagonal line across one face.

Each set of four values is in the order: plot or draw value, x co-ordinate, y co-ordinate, z co-ordinate. The values are calculated by mentally tracing the outline of the object in three dimensions. Using an imaginary pen, visit each corner point of the object's outline in turn. If your pen is moved to a point without drawing a line, 4 is used for the first value. A value of 5 indicates that a line is to be drawn to the point from the preceding one. These particular values are used because they make the program for the BBC Micro less complicated.

The co-ordinate origin (0,0) is at the centre of the screen. It is best to make this the centre of your object. The x axis is the horizontal axis, with positive values going upwards. The z axis is the axis in and out of the screen. The positive direction on this axis is into the screen.

Keep the X, Y, and Z values as small as is reasonably possible. The initial setting of the perspective effect and viewpoint distance must take into account that an object width of about 10 will fill the screen. This could be changed by altering the scaling factor used in the perspective transformation. We suggest that you experiment with simple shapes first. Keep the number of points low. When you have mastered the digitising of simple solid objects (a pyramid, a cube) you can move on to more complex ones.

Our program could be extended further to give additional effects. A translation facility could be

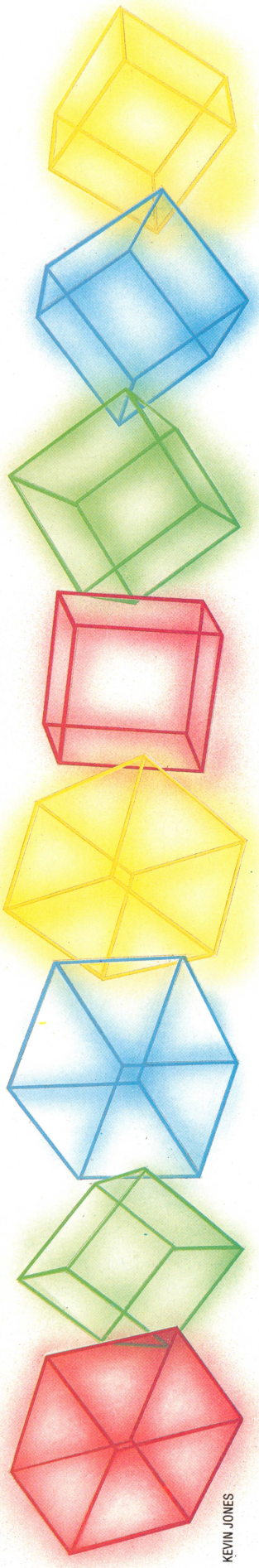
incorporated to move the object, without rotation, relative to the co-ordinate origin. You could attempt to incorporate a routine to remove the lines and parts of lines that should be hidden from view. This makes the present wire frame image much more realistic. However, such hidden line removal is a vastly complicated matter. It requires very complex mathematics and would slow the program down a great deal. Even if you never add to this program but leave it exactly as it is given here, you can still achieve some pretty spectacular, or just pretty, results.

Spectrum Version

```
 10 LET N=16
 20 DIM P(50)
 21 DIM X(50)
 22 DIM Y(50)
 23 DIM Z(50)
 24 DIM A(50)
 25 DIM B(50)
 40 LET D=10: LET P=0.5
 50 LET SI=SIN 0.09: LET CO=COS 0.09
 60 FOR I=1 TO N
 70 READ P(I),X(I),Y(I),Z(I)
 80 NEXT I
 90:
200 INVERSE 0: GO SUB 300
210 IF INKEY$<>"" THEN  GO TO 210
211 IF INKEY$="" THEN  GO TO 211
212 LET I$=INKEY$
230 INVERSE 1: GO SUB 300
240 IF I$="1" THEN  GO SUB 1000
241 IF I$="2" THEN  GO SUB 2000
242 IF I$="3" THEN  GO SUB 3000
243 IF I$="4" THEN  GO SUB 4000
244 IF I$="5" THEN  GO SUB 5000
245 IF I$="6" THEN  GO SUB 6000
246 IF I$="7" THEN  GO SUB 7000
247 IF I$="8" THEN  GO SUB 8000
250 GO TO 200
260:
300 FOR I=1 TO N
310 LET A(I)=X(I)*300/(P*Z(I)+D): LET B(I)=Y(
I)*300/(P*Z(I)+D)
320 NEXT I
330 FOR I=1 TO N
340 IF P(I)=4 THEN  PLOT A(I)+128,B(I)+85
345 IF P(I)=5 THEN  DRAW A(I)-A(I-1),B(I)-B(I
-1)
350 NEXT I
360 RETURN
370:
1000 FOR I=1 TO N
1010 LET X=X(I)*CO-Z(I)*SI
1020 LET Z=Z(I)*CO+X(I)*SI
1030 LET X(I)=X: LET Z(I)=Z
1040 NEXT I
1050 RETURN
1060:
2000 FOR I=1 TO N
2010 LET X=X(I)*CO+Z(I)*SI
2020 LET Z=Z(I)*CO-X(I)*SI
2030 LET X(I)=X: LET Z(I)=Z
2040 NEXT I
2050 RETURN
2060:
3000 FOR I=1 TO N
3010 LET Y=Y(I)*CO+Z(I)*SI
3020 LET Z=Z(I)*CO-Y(I)*SI
3030 LET Y(I)=Y: LET Z(I)=Z
3040 NEXT I
3050 RETURN
3060:
4000 FOR I=1 TO N
4010 LET Y=Y(I)*CO-Z(I)*SI
4020 LET Z=Z(I)*CO+Y(I)*SI
4030 LET Y(I)=Y: LET Z(I)=Z
4040 NEXT I
4050 RETURN
4060:
5000 LET D=D*0.9
5010 RETURN
5020:
6000 LET D=D/0.9
6010 RETURN
6020:
7000 LET P=P/0.9
7010 RETURN
7020:
8000 LET P=P*0.9
8010 RETURN
8020:
9000 DATA 4,1,1,1, 5,1,1,-1, 5,-1,1,-1, 5,-1,1
,1,5,1,1,1
9010 DATA 5,1,-1,1, 5,1,-1,-1, 5,-1,-1,-1, 5,-
1,-1,1, 5,1,-1,1
9020 DATA 4,1,-1,-1, 5,1,1,-1, 5,-1,-1,-1, 5,-
1,1,-1
9030 DATA 4,-1,1,1, 5,-1,-1,1
```

BBC Version

```
 10 N=16
 20 DIM P(50),X(50),Y(50),Z(50),A(50),B(50)
 30 MODE0:VDU29,640;512;5
 40 D=10:P=0.5
 50 SI=SIN(0.09):CO=COS(0.09)
 60 FOR I=1 TO N
 70   READ P(I),X(I),Y(I),Z(I)
 80   NEXT I
 90 :
200 GCOL0,3:GOSUB 300
210 I$=GET$
220 V=VAL(I$)
230 GCOL0,0:GOSUB 300
240 ON V GOSUB 1000,2000,3000,4000,5000,6000
,7000,8000 ELSE 150
250 GOTO 200
260 :
300 FOR I=1TON
310   A(I)=X(I)*1000/(P*Z(I)+D):B(I)=Y(I)*
1000/(P*Z(I)+D)
320   NEXT I
330 FOR I=1 TO N
340   PLOT P(I),A(I),B(I)
350   NEXT I
360 RETURN
370 :
1000 FOR I=1 TO N
1010   X=X(I)*CO-Z(I)*SI
1020   Z=Z(I)*CO+X(I)*SI
1030   X(I)=X:Z(I)=Z
1040   NEXT I
1050 RETURN
1060 :
2000 FOR I=1 TO N
2010   X=X(I)*CO+Z(I)*SI
2020   Z=Z(I)*CO-X(I)*SI
2030   X(I)=X:Z(I)=Z
2040   NEXT I
2050 RETURN
2060 :
3000 FOR I=1 TO N
3010   Y=Y(I)*CO+Z(I)*SI
3020   Z=Z(I)*CO-Y(I)*SI
3030   Y(I)=Y:Z(I)=Z
3040   NEXT I
3050 RETURN
3060 :
4000 FOR I=1 TO N
4010   Y=Y(I)*CO-Z(I)*SI
4020   Z=Z(I)*CO+Y(I)*SI
4030   Y(I)=Y:Z(I)=Z
4040   NEXT I
4050 RETURN
4060 :
5000 D=D*.9
5010 RETURN
5020 :
6000 D=D/.9
6010 RETURN
6020 :
7000 P=P/.9
7010 RETURN
7020 :
8000 P=P*.9
8010 RETURN
8020 :
10000 DATA 4,1,1,1, 5,1,1,-1, 5,-1,1,-1, 5,-1,
1,1,5,1,1,1
10010 DATA 5,1,-1,1, 5,1,-1,-1, 5,-1,-1,-1, 5,
-1,-1,1, 5,1,-1,1
10020 DATA 4,1,-1,-1, 5,1,1,-1, 5,-1,-1,-1, 5,
-1,1,-1
10030 DATA 4,-1,1,1, 5,-1,-1,1
```



KEVIN JONES

PIECES OF THE PUZZLE

The most efficient way to create programs in any language is to use 'modular structuring'. Some languages, such as PASCAL, encourage this approach, while BASIC users need to discipline themselves to adopt the technique. We show you how your programming will be greatly enhanced using modules of code as your basic components.

A module is a piece of code that performs a particular function. The points of entry and exit, known as the module's 'interfaces', must be precisely defined, and the processes that occur between these interfaces should be entirely independent of the rest of the program. Once a module has been written, it can be treated as a 'black box'. Data may pass in and out of the module's interfaces, but what goes on inside can be left to itself.

Modules can be joined together to build up a program without the writer having to worry about how they perform their tasks. A stock of modules can be built up by a programmer to be used when needed, and programmers can pass modules on to be used in another writer's programs. But in order to take advantage of modular structured programming, we need to take careful note of the flow of control and the flow of data when we write the modules.

To ensure that all your modules behave in the same way with respect to the flow of control, a very simple rule should be followed: all modules should have a single entry point and a single exit. What this means in practice is that the flow of control within the module has to be carefully designed so that it starts at one place and, no matter how much it loops and branches, it reaches the same exit by all possible routes.

Modules correspond to the algorithms we have been looking at in previous instalments of the course. 'Structured' languages, such as PASCAL, allow the programmer to create subroutines that may be called by name, and which use their own variables. Such languages encourage a programmer to enter or leave a routine (called a 'procedure') by single entry and exit points.

In BASIC, using the GOSUB . . . RETURN combination, a subroutine can be called from the main program and, after the subroutine has been carried out, control will return to the line immediately after the GOSUB command. However, there is no restriction on which line the GOSUB sends control to. Two different GOSUBs may send control to different lines of a subroutine with a single RETURN, and the result might be completely different in each case. Similarly, there is no restriction on how many RETURN statements may be used in a subroutine.

This means that the BASIC programmer must be self-disciplined. You should start by making sure that all GOSUBs to the same subroutine point to the same line number, and that every subroutine has only one RETURN in it. It is best to get in the habit of marking the first line of each subroutine with a REM statement giving it a title, and use that line as the entry point. Make the RETURN the last line of the subroutine. This is not essential but it makes things much clearer.

THE GOTO RULE

Extra care should be taken with the GOTO command, which can play havoc with program structure. The rule here is: only use a GOTO to send control to a line *within* the same subroutine. This avoids the potential danger of skipping over a RETURN or passing control to the wrong RETURN. There are times when it is necessary to leave a routine without executing every line. In this case you should GOTO the line with RETURN on it, and there should be no problems.

Using GOTO within loops is even more dangerous. If control jumps out of a loop, BASIC cannot know this and assumes that the rest of the program is the body of that loop! The safety rule is: when in the body of a loop, never GOTO a line outside the body of that loop. If a loop needs to be terminated early, set the loop counter or test variable to the terminal value and GOTO the test line (the line with NEXT or WHILE in it). As with the RETURN statement, put NEXT or WHILE on a line of its own to make this easier. Keeping track of the structure of a program is a lot simpler if GOTOs are avoided as much as possible.

Branches are the most likely place for control to go astray, so try not to allow any decisions to send control out of a subroutine unless it is with a proper call to another subroutine. Remember that each subroutine has a single exit point, so make sure that it is possible to follow the flow of control through every branch to that point. Drawing a flow chart for the routine makes this easy to check. Setting a flag can often reduce the need for GOTOs in routines involving loops and branches.

We can think of data passing in and out of modules, just as we did for algorithms (see page 386). So that modules can be used independently of each other, you must design them so that the only influence they have on each other is through the data that passes between them. The main program passes data to a module and, when the module has been executed, any result that has

been generated is passed back.

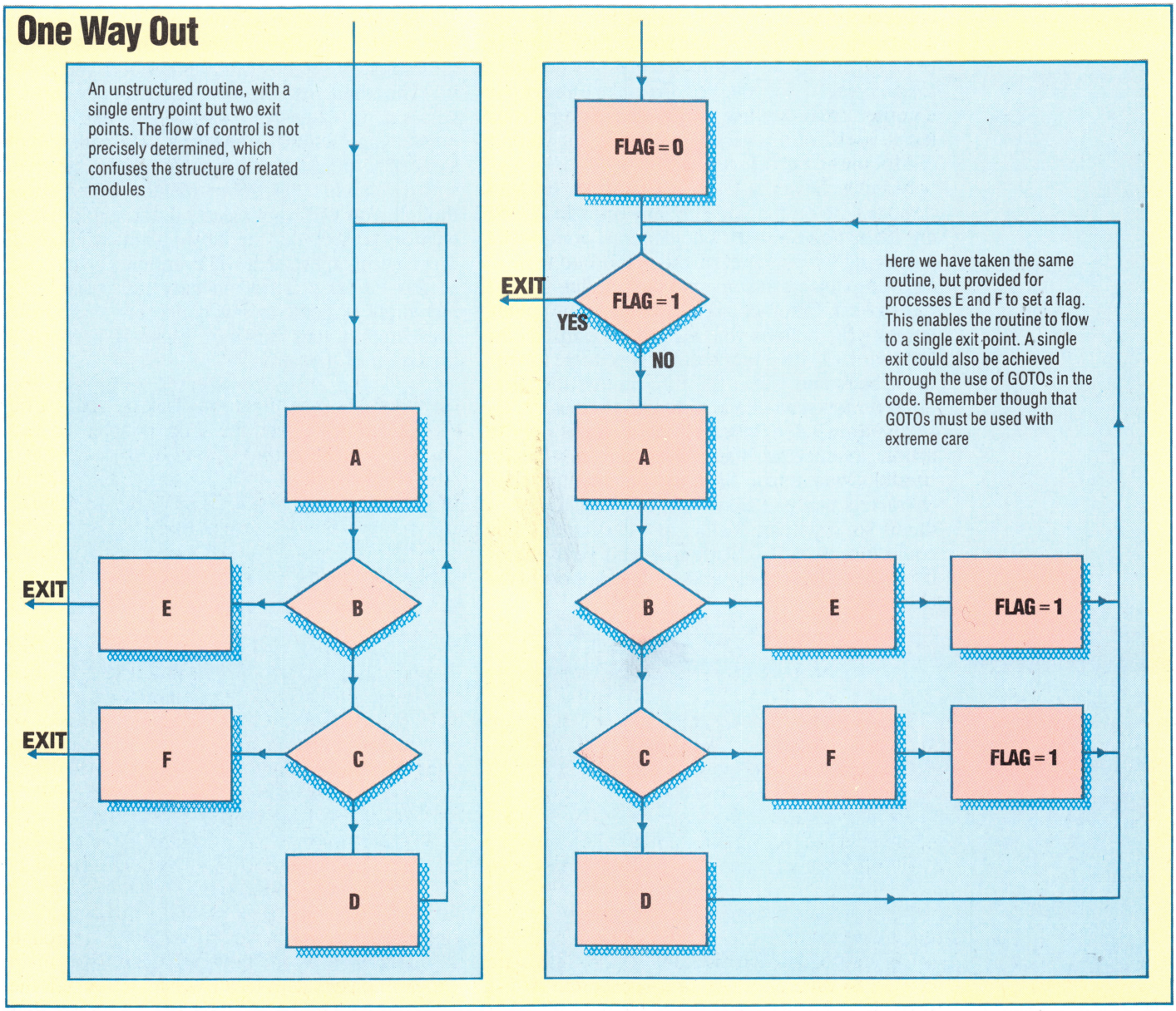
Data moves around programs inside variables and the freedom of movement of a variable is called its 'scope'. Many programming languages can restrict the scope of a variable to particular subroutines. In PASCAL, the variables used in a particular subroutine (procedure) must be 'declared' for that procedure. Variables declared for the main program are *global* and may be used anywhere in the program (including within any of its modules). Variables declared within a particular procedure, however, are *local* to that procedure and can only be used there.

Local variables can have the same names as global ones and using one does not affect the value of the other. Using a language that supports local variables allows us to write subroutines without having to worry about how the variables used in the routine might affect variables in other routines. Unfortunately, very few versions of the BASIC language support local variables, which means that if we wish to write independent subroutines we must somehow simulate the effect of having local variables.

The simplest way to do this is to adopt naming conventions that distinguish variables that do different jobs. Some conventions already exist and are widely used by programmers. Using I, J and K as loop counters and index values is very common, a practice that has been adopted from mathematics.

Having described a program with a flowchart, it is a simple matter to number the subroutines involved, or to give them some other kind of code. Any global variables that need to be made local to a particular subroutine can then be suffixed by this code to make them unique. Thus, routine number 5 may use the local variables SUM5 and TOTAL5 to distinguish them from SUM12 and TOTAL12 in routine number 12. Be careful though that the BASIC you are using doesn't look only at the first two characters! Variables that are used to pass values between subroutines and those used only in the main program need not be coded.

One Way Out



LIZ DIXON

PLAYING WITH THE GODS

Valhalla is an adventure game flavoured with references to Nordic mythology. Written for the Spectrum and Commodore 64, it has proved hugely successful, capturing the public's imagination through its innovative programming and spectacular graphics. It has won for Legend, its manufacturers, several software awards.

There are 81 locations in the Valhalla adventure, of which 16 are in Asgard — including the elusive Valhalla, the place of your dreams. Midgard consists of 20 locations, and the remaining 45 collectively make up Hell. There are eight exits from each location, although some of these may be blocked or require you to have possession of a magic object to let you pass. To make things more complex, there are 'ringways' connecting distant locations; if you are wearing a suitable ring you can 'jump' to the far-off scene.

Valhalla has a cast of 36 characters (at least three will be on the screen at any one time), who are either good or bad. They interact continually, having fights, giving each other food or wine or throwing objects at someone they dislike.

You can interact with them at any point, entering the actions you wish to perform via the keyboard. On the other hand, it is possible for you to sit back and watch the fun, though the silent observer types will invariably end up dead.

The game does not present a single path of action to success; the player can take many possible routes. You have to convince the good characters that you are worthy of being helped by them. So, if you are feeling in a dark mood, you could just as easily decide to spend your energy being thoroughly nasty and get the support of the bad guys.

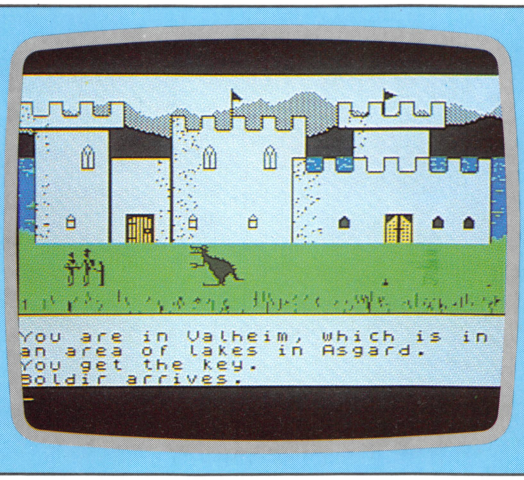
The graphic displays of Valhalla are particularly good, especially on the Commodore version. A background for each location is first drawn on screen, in vivid primary colours on the Spectrum and subtle pastel hues on the Commodore. Next the characters appear: on the Spectrum, these are rather stick-like and drawn in black, while the Commodore's greater colour potential and higher resolution allow for finer character detail. Finally, the objects present at the location appear: these could include food, wine, jewels, keys and weapons. Once again, the Spectrum's representations are far less realistic than those of the Commodore version.

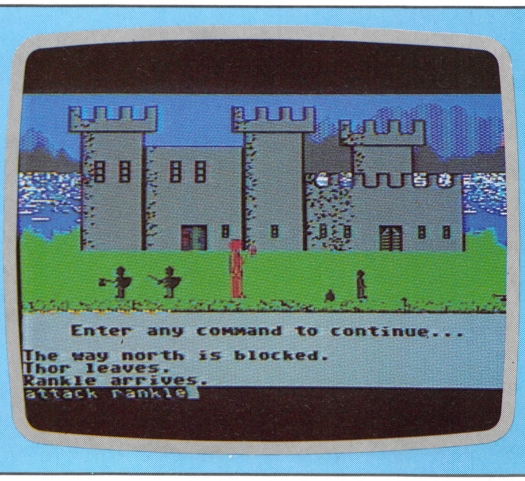
The characters then interact with each other, and their actions are described in words at the bottom of the screen. And this is where you come in. The range of actions that the player can perform is extensive: you might like to try persuading one of the Nordic Gods to part with some of his treasure, or you could try attacking him with a weapon to test your strength. After a while, though, you will want to go off and explore new territory to seek out the fabled Valhalla. Finding the magical objects that will eventually give access to this Paradise is not an easy task; many are incredibly difficult to lay hands on — and the exasperation that this can cause is perhaps a limitation of the game.

Valhalla: For 48K Spectrum, £14.95
For Commodore 64, £14.95
Publishers: Legend, Freepost, 1 Milton Road, Cambridge CB4 1UY
Authors: Graham Asher, Richard Edwards, Charles Goodwin, James Learmont, Jan Ostler, Andrew Owen, John Peel
Joysticks: Not required
Format: Cassette

Day For Night

Valhalla is one of several games produced recently whose graphics indicate the passage of time by darkening the screen. This creates the effect of 'day' and 'night'





LIZ HEANEY

THE RING CYCLE

So far in the course we have built up subroutines to make use of the high resolution capabilities of the Commodore 64. These have included Plotsub and Linesub. Here, we look at a routine that will draw circles, given the centre co-ordinates and radius.

It is impossible to produce a precisely drawn circle on a home computer. The accuracy of the approximation we can make depends on the method used and the length and complexity of the final routine. Circle drawing from BASIC usually involves calculations using either the sine and cosine functions or squares and square roots to produce the co-ordinates of points on the circumference of the circle to be drawn. Both of these methods, however, produce difficulties when we try to implement them in machine code, so let's look at an alternative method, which is particularly suited to a machine code solution.

The method we shall use considers the diameter of a circle to be divided into an equal number of parts, each of width W. At each division we can think of a rod that reaches vertically upwards to a point C, on the circumference of the circle. The diagram shows one such rod, N divisions from the left-hand end of the diameter, AB. By joining A and B to C we form two right-angled triangles, ACD and BCD, as shown on the right.

Using Pythagoras's theorem, we can write down the following expressions from this diagram:

$$AC^2 = AD^2 + CD^2$$
$$CB^2 = DB^2 + CD^2$$

If we add these equations together we get:

$$AC^2 + CB^2 = AD^2 + DB^2 + 2CD^2$$

However, it is a special property of circles that triangle ABC is also right-angled. So, we can say:

$$AC^2 + CB^2 = AB^2$$

Putting this into the left-hand side of the earlier equation we get:

$$AB^2 = AD^2 + DB^2 + 2CD^2$$

CD corresponds to 'yup' — the distance from the diameter to the circumference. AD is N×W and AB is 2×R, where R is the radius of the circle. Substituting these values into the equation and rearranging these factors we get:

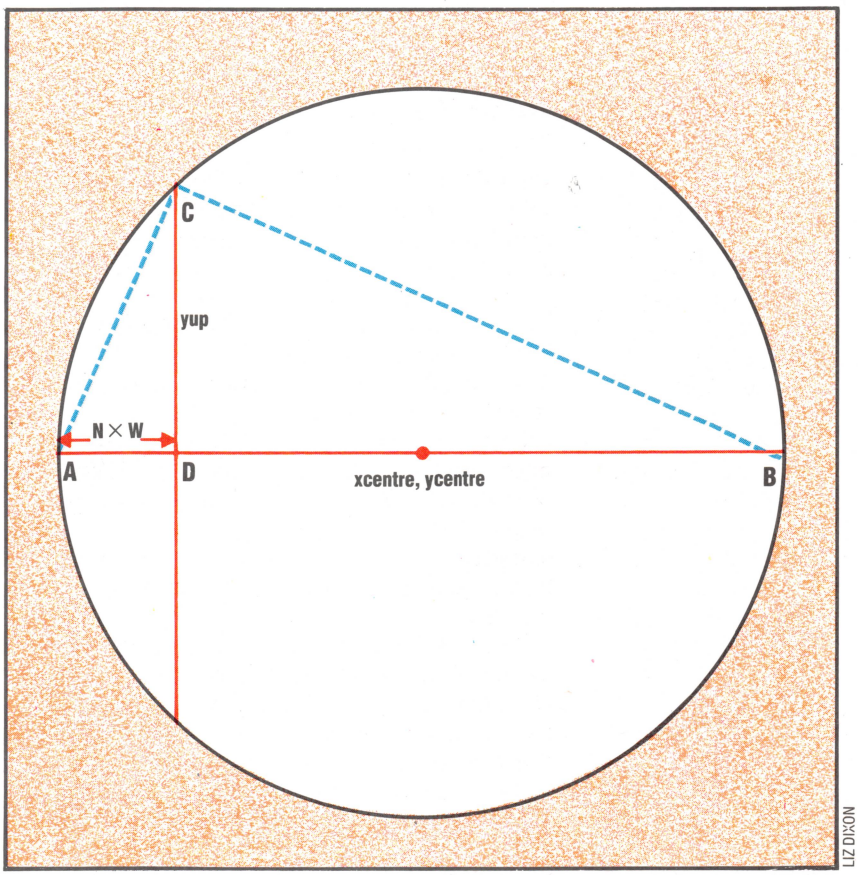
$$2yup^2 = (2R)^2 - (NW)^2 - (2R-NW)^2$$
$$yup^2 = 2RNW - N^2 W^2$$

If we decide to divide the diameter into 64 equal parts, then $W = 2\times R/64$, which reduces to $W = R/32$. Substituting this into the equation:

$$yup^2 = 2RNR/32 - N^2 R^2/32^2$$
$$yup^2 = R^2/32^2 \times(64N-N^2)$$

and finding the square root of both sides gives:

$$yup = R/32\times SQR(64N - N^2)$$

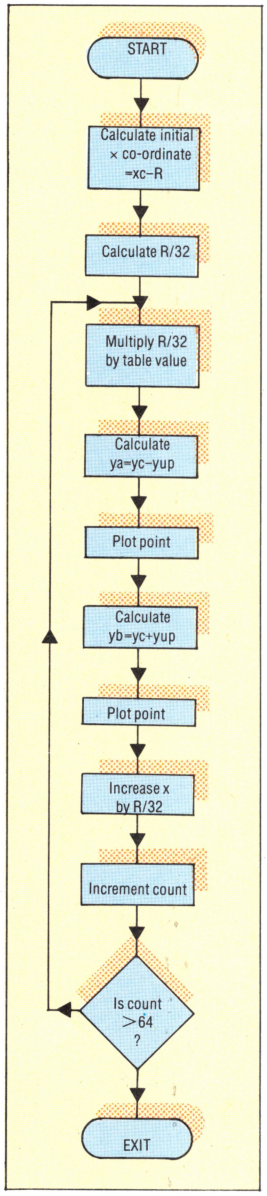


If we start with x = xcentre − R and increment in 64 equal steps, then at each increment the distance up to the circumference will be given by the formula we have arrived at, where N is the step number. Although this result includes a square root, we should note that the expression to be square-rooted is independent of the centre co-ordinates or radius of the circle. We can, therefore, calculate a table of values for the square root function, which gives a solution for each value of N from 0 to 64. This needs to be done only once and can be incorporated into our program as a 'look-up' table.

The absolute y co-ordinate for each increment of x is:

$$ya = ycentre - yup$$

We can also make use of the symmetry of the circle



to calculate the corresponding y co-ordinate in the bottom half of the circle:

yb = ycentre + yup

DEVELOPING OUR ROUTINE

We can now map out the structure of the routine in detail. The flowchart shows how each point on the circumference can be calculated. We can see that the routine should be fairly fast as only one multiplication needs to be performed for each point plotted. However, there are two slight problems with the routine as it stands. First of all, we will not produce a continuous circle, only a series of points around the circumference. Secondly, although this technique produces well-defined circles when used from BASIC, there are inaccuracies when it is done in machine code.

The first problem can be solved by making use of Linesub (see page 419) to join the points with short lines, to produce a continuous circle. The second problem is due to inaccuracies in the calculation of the square root 'look-up' table. Calculating the values in BASIC and POKEing them into a series of bytes set aside for the table means that only the integer part of each value is actually stored. To get more well-defined circles, we must improve the accuracy of the values stored in the table. The maximum number to be stored is 32, and therefore we can multiply each number by eight before storing it, and still use only one byte per table entry. 32 is the only number, when multiplied by eight, which can't be held in a single byte. To simplify the routine we'll approximate this value. Our machine code routine can then divide the table value by eight, by performing three LSRs (logical shifts right) on it. More importantly, it can keep the remainder after division for use in further calculations.

The source code listing sets aside 65 bytes at the beginning of the program to store the table, the first byte of which is labelled. Subsequent entries in the table can be accessed by indexed addressing. The source code listing can be entered and assembled into memory as usual. However, before SAVEing the assembled code, the following BASIC program should be entered and RUN. It will not corrupt your object code, as this is located high in memory. The program calculates the 65 table values required by our Circsub program, multiplies these values by eight and POKEs the result into the area of memory set aside in your machine code program. Once the table creation program has been RUN the machine code can be SAVEd in the usual way, but ensure that you SAVE the table area in with the object code. The table starts at $C500. Once this has been done, the 'look-up' table will be loaded automatically whenever you load Circsub.

```
5 REM**** CREATE CIRCSUB TABLE ****
10 FORN=0TO64
20 X=SQR(64*N-N↑2)*8
25 X%=X:DF=X-X%
27 IFDF>=.5THENX%=X%+1
28 IFX%>255THENX%=255
29 POKE50432+N,X%
30 NEXT
```

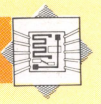
The following program shows how Circsub can be used from within a BASIC program. All that is required is to specify the centre co-ordinates and the radius. The subroutine at line 2000 splits the x co-ordinate into LO-byte/HI-byte form, then POKEs the specified values to Circsub and makes the appropriate SYS call. Note that, as Circsub uses Linesub, which in turn uses Plotsub (see page 337), all three subroutines must be loaded at the start of the program. This program draws circles with increasing radii across the screen.

```
3 REM**** CIRCSUB TEST PROGRAM ****
5 DN=8:REM FOR CASSETTE DN=1
10 IFA=0THENA=1:LOAD"PLOTSUB.HEX",DN,1
15 IFA=1THENA=2:LOAD"LINESUB.HEX",DN,1
20 IFA=2THENA=3:LOAD"CIRCSUB.HEX",DN,1
100 GOSUB1000
110 YC=100:R=2
120 FORXC=30TO210STEP7
130 R=R+3
140 GOSUB2000
150 NEXT
160 GETA$:IFA$=""THEN160
170 GOSUB3000
180 END
1000 REM **** SET HIRES ****
1010 POKE49408,1:POKE49409,1
1020 POKE49410,3
1030 SYS49422
1040 RETURN
1050 :
2000 REM **** ENTER CIRCSUB ****
2010 CHI=INT(XC/256):CLO=XC-256*CHI
2020 POKE50497,CLO:POKE50498,CHI
2030 POKE50499,YC
2040 POKE50500,R
2050 SYS50521
2060 RETURN
2070 :
3000 REM **** CLEAR HIRES ****
3005 RESTORE
3007 PRINTCHR$(147):REM CLEAR SCREEN
3008 PRINTTAB(10)"CIRCSUB VARIABLES"
3009 PRINT
3010 POKE49408,0:SYS49422
3030 FORI=50497TO50497+23STEP2
3035 READA$
3040 PRINTTAB(2);A$,PEEK(I),
3045 READA$
3047 PRINTA$,PEEK(I+1)
3050 NEXT
3060 RETURN
3070 :
5000 DATAXCTRLO,XCTRHI,YCTR,RADIUS,INTR
5010 DATAREMR,TOTX,RESREM,RESLO,RESHI,OLDXLO
5020 DATAOLDXHI,NEWXLO,NEWXHI,OLDYA
5030 DATAOLDYB,NEWYA,NEWYB,XFLAG,YFLAG,OLDY,NEWY
5040 DATAINTTAB,REMTAB
```

As with the other high resolution subroutines for the Commodore 64, the machine code can also be entered as a series of DATA statements if you do not have an assembler. The listing below should be typed and RUN to load Circsub into memory. Note that the BASIC loaders for Linesub and Circsub should also be loaded and RUN prior to loading the demonstration program. As the three routines will now be in memory, lines 10, 15 and 20 of the demo program should be omitted. Note that the BASIC loader for Circsub already contains the look-up table data and so it is not necessary to RUN the 'create table' program in this case.

Coming Soon

The machine code section of The Home Computer Advanced Course has so far concentrated on home computers with the Z80 and 6502 microprocessor. But there is a third popular microprocessor, the 6809 chip, which is used in the Dragon and Tandy Color computers. The course will soon be turning its attention to this chip and teaching 6809 owners to program in machine code too



BASIC Loader Program

```
10 REM**** BASIC LOADER FOR CIRCSUB ****
20 FORI=50432TO50432+498
30 READA:POKEI,A
40 CC=CC+A
50 NEXT I
60 READA:IFA<>CCTHENPRINT"CHECKSUM ERROR"
100 DATA0,63,89,108,123,137,149,159
110 DATA169,177,185,193,199,205,211
120 DATA216,221,226,230,233,237,240
130 DATA243,245,247,249,251,252,253
140 DATA254,255,255,255,255,255,254
150 DATA253,252,251,249,247,245,243
160 DATA240,237,233,230,226,221,216
170 DATA211,205,199,193,185,177,169
180 DATA159,149,137,123,108,89,63,0
190 DATA205,0,100,80,2,16,16,0,0,0,29
200 DATA1,31,1,100,100,100,100,0,0,120
210 DATA100,0,0,72,138,72,152,72,173
220 DATA68,197,41,31,141,70,197,173,68
230 DATA197,160,5,74,136,208,252,141
240 DATA69,197,173,65,197,56,237,68
250 DATA197,141,77,197,141,75,197,173
260 DATA66,197,233,0,141,78,197,141,76
270 DATA197,173,67,197,141,79,197,141
280 DATA80,197,169,0,170,141,71,197
290 DATA189,0,197,160,3,74,136,208,252
300 DATA141,87,197,189,0,197,41,7,141
310 DATA88,197,172,68,197,169,0,141,72
320 DATA197,141,73,197,141,74,197,173
330 DATA72,197,24,109,88,197,141,72
340 DATA197,201,8,144,23,56,233,8,141
350 DATA72,197,173,73,197,24,105,1,141
360 DATA73,197,173,74,197,105,0,141,74
370 DATA197,173,73,197,24,109,87,197
380 DATA141,73,197,173,74,197,105,0
390 DATA141,74,197,136,208,198,160,5
400 DATA78,74,197,110,73,197,110,72
410 DATA197,136,208,244,173,72,197,201
420 DATA16,144,9,173,73,197,24,105,1
430 DATA141,73,197,173,67,197,56,237
440 DATA73,197,141,81,197,173,81,197
450 DATA141,86,197,173,79,197,141,85
460 DATA197,32,165,198,173,67,197,24
470 DATA109,73,197,141,82,197,173,82
480 DATA197,141,86,197,173,80,197,141
490 DATA85,197,32,165,198,173,77,197
500 DATA141,75,197,173,78,197,141,76
510 DATA197,173,81,197,141,79,197,173
520 DATA82,197,141,80,197,173,71,197
530 DATA24,109,70,197,141,71,197,201
540 DATA32,144,26,173,71,197,56,233,32
550 DATA141,71,197,173,77,197,24,105,1
560 DATA141,77,197,173,76,197,105,0
570 DATA141,78,197,173,77,197,24,109
580 DATA69,197,141,77,197,173,78,197
590 DATA105,0,141,78,197,232,224,65
600 DATA240,3,76,153,197,104,168,104
610 DATA170,104,96,169,0,141,83,197
620 DATA141,84,197,173,75,197,205,77
630 DATA197,208,3,238,83,197,173,85
640 DATA197,205,86,197,208,3,238,84
650 DATA197,173,83,197,45,84,197,208
660 DATA39,173,75,197,141,0,195,173,76
670 DATA197,141,1,195,173,85,197,141,4
680 DATA195,173,77,197,141,2,195,173
690 DATA78,197,141,3,195,173,86,197
700 DATA141,5,195,32,14,195,96
710 DATA67223:REM*CHECKSUM*
```

Circsub Program

```
;++++++++++++++++++
;++++++++++++++++++
;++              ++
;++  CIRCSUB 64  ++
;++              ++
;++++++++++++++++++
;++++++++++++++++++
;++++ LINESUB VARIABLES ++++
;
LINSUB = $C30E
X1LO   = $C300
X1HI   = $C301
X2LO   = $C302
X2HI   = $C303
Y1     = $C304
Y2     = $C305
*      = $C500
;
;++++ CIRCSUB VARIABLES ++++
;
TABLE  *=*+65          ; LOOK UP TABLE
XCTRLO *=*+1
XCTRHI *=*+1
YCTR   *=*+1
RADIUS *=*+1
INTR   *=*+1
REMR   *=*+1
TOTX   *=*+1
RESREM *=*+1
RESLO  *=*+1
RESHI  *=*+1
OLDXLO *=*+1
OLDXHI *=*+1
NEWXLO *=*+1
NEWXHI *=*+1
OLDYA  *=*+1
OLDYB  *=*+1
NEWYA  *=*+1
NEWYB  *=*+1
XFLAG  *=*+1
YFLAG  *=*+1
OLDY   *=*+1
NEWY   *=*+1
INTTAB *=*+1
REMTAB *=*+1
;
;++++ PUSH REGISTERS ONTO STACK ++++
;
       PHA
       TXA
       PHA
       TYA
       PHA
;
;++++ CALC INT(R/32) AND REMR ++++
;
AD 44 C5       LDA RADIUS
29 1F          AND #$1F
8D 46 C5       STA REMR
AD 44 C5       LDA RADIUS
A0 05          LDY #$05
4A       LOOP1 LSR A
88             DEY
D0 FC          BNE LOOP1
8D 45 C5       STA INTR
;
;++++ CALC INITIAL X COORDINATE++++
;
AD 41 C5       LDA XCTRLO
38             SEC
ED 44 C5       SBC RADIUS
8D 4D C5       STA NEWXLO
8D 4B C5       STA OLDXLO
AD 42 C5       LDA XCTRHI
E9 00          SBC #$00
8D 4E C5       STA NEWXHI
8D 4C C5       STA OLDXHI
;
;++++ SET INITIAL OLD Y VALUES ++++
;
AD 43 C5       LDA YCTR
8D 4F C5       STA OLDYA
8D 50 C5       STA OLDYB
;
;++++ ZERO COUNT AND TOTX ++++
;
A9 00          LDA #$00
AA             TAX
8D 47 C5       STA TOTX
;
;++++ CALC INTTAB AND REMTAB ++++
;
NEXTPT
BD 00 C5       LDA TABLE,X
A0 03          LDY #$03
       LOOP2
4A             LSR A
88             DEY
D0 FC          BNE LOOP2
8D 57 C5       STA INTTAB
;
BD 00 C5       LDA TABLE,X
29 07          AND #$07
8D 58 C5       STA REMTAB
;
;++++ MULTIPLY TABLE BY RADIUS ++++
;
AC 44 C5       LDY RADIUS
A9 00          LDA #$00
8D 48 C5       STA RESREM
8D 49 C5       STA RESLO       ;ZERO ANSWER
8D 4A C5       STA RESHI
;
AD 48 C5 AGAIN LDA RESREM
18             CLC
6D 58 C5       ADC REMTAB
8D 48 C5       STA RESREM
C9 08          CMP #$08        ;>= 8?
90 17          BCC NOCARR
38             SEC
E9 08          SBC #$08        ;RESET REM
8D 48 C5       STA RESREM
AD 49 C5       LDA RESLO
18             CLC
69 01          ADC #$01        ;INC RESULT
8D 49 C5       STA RESLO
AD 4A C5       LDA RESHI
69 00          ADC #$00
8D 4A C5       STA RESHI
       NOCARR
AD 49 C5       LDA RESLO
18             CLC
6D 57 C5       ADC INTTAB      ; ADD INTEGER PART
8D 49 C5       STA RESLO
AD 4A C5       LDA RESHI
69 00          ADC #$00
8D 4A C5       STA RESHI
88             DEY
D0 C6          BNE AGAIN
;
;++++ DIVIDE RESULT BY 32 ++++
;
A0 05          LDY #$05
       LOOP3
4E 4A C5       LSR RESHI
6E 49 C5       ROR RESLO
6E 48 C5       ROR RESREM
88             DEY
D0 F4          BNE LOOP3
;
AD 48 C5       LDA RESREM
C9 10          CMP #$10        ; REM >=16?
90 09          BCC NOCRRY
AD 49 C5       LDA RESLO
18             CLC
69 01          ADC #$01        ; ADD 1
8D 49 C5       STA RESLO
;
;++++ CALC 1ST Y COORDINATE ++++
;
       NOCRRY
AD 43 C5       LDA YCTR
38             SEC
ED 49 C5       SBC RESLO
8D 51 C5       STA NEWYA
;
; ++++ DRAW LINE ++++
;
AD 51 C5       LDA NEWYA
8D 56 C5       STA NEWY
AD 4F C5       LDA OLDYA
8D 55 C5       STA OLDY
20 A5 C6       JSR DRAW
;
;++++ CALC 2ND Y COORDINATE ++++
;
AD 43 C5       LDA YCTR
18             CLC
6D 49 C5       ADC RESLO
8D 52 C5       STA NEWYB
;
;++++ DRAW LINE ++++
;
AD 52 C5       LDA NEWYB
8D 56 C5       STA NEWY
AD 50 C5       LDA OLDYB
8D 55 C5       STA OLDY
20 A5 C6       JSR DRAW
;
;++++ SWOP OLD FOR NEW ++++
;
AD 4D C5       LDA NEWXLO
8D 4B C5       STA OLDXLO
AD 4E C5       LDA NEWXHI
8D 4C C5       STA OLDXHI
AD 51 C5       LDA NEWYA
8D 4F C5       STA OLDYA
AD 52 C5       LDA NEWYB
8D 50 C5       STA OLDYB
;
;++++ ALTER X COORDINATE ++++
;
AD 47 C5       LDA TOTX
18             CLC
6D 46 C5       ADC REMR
8D 47 C5       STA TOTX        ;ADD REMR TO TOTAL
C9 20          CMP #$20        ;>=32?
90 1A          BCC NOINC
AD 47 C5       LDA TOTX
38             SEC
E9 20          SBC #$20        ;RESET TOTX
8D 47 C5       STA TOTX
AD 4D C5       LDA NEWXLO
18             CLC
69 01          ADC #$01        ; INC X COORD
8D 4D C5       STA NEWXLO
AD 4C C5       LDA OLDXHI
69 00          ADC #$00
8D 4E C5       STA NEWXHI
       NOINC
AD 4D C5       LDA NEWXLO
18             CLC
6D 45 C5       ADC INTR
8D 4D C5       STA NEWXLO
AD 4E C5       LDA NEWXHI
69 00          ADC #$00
8D 4E C5       STA NEWXHI
;
;++++ INCREMENT COUNTER ++++
;
E8             INX
E0 41          CPX #$41
F0 03          BEQ FINISH
4C 99 C5       JMP NEXTPT
;
;++++ PULL REGISTERS OFF STACK ++++
;
FINISH
68             PLA
A8             TAY
68             PLA
AA             TAX
68             PLA
60             RTS
;
;++++ CHECK FOR SAME POINT ++++
;
DRAW
A9 00          LDA #$00
8D 53 C5       STA XFLAG       ;ZERO FLAGS
8D 54 C5       STA YFLAG
;
AD 4B C5       LDA OLDXLO
CD 4D C5       CMP NEWXLO
D0 03          BNE NOXFLG
       NOXFLG
AD 55 C5       LDA OLDY
CD 56 C5       CMP NEWY
D0 03          BNE NOYFLG
EE 54 C5       INC YFLAG
       NOYFLG
AD 53 C5       LDA XFLAG
2D 54 C5       AND YFLAG
D0 27          BNE NODRAW
;
; ++++ DRAW LINE ++++
;
AD 4B C5       LDA OLDXLO
8D 00 C3       STA X1LO
AD 4C C5       LDA OLDXHI
8D 01 C3       STA X1HI
AD 55 C5       LDA OLDY
8D 04 C3       STA Y1
AD 4D C5       LDA NEWXLO
8D 02 C3       STA X2LO
AD 4E C5       LDA NEWXHI
8D 03 C3       STA X2HI
AD 56 C5       LDA NEWY
8D 05 C3       STA Y2
;
20 0E C3       JSR LINSUB
;
       NODRAW
60             RTS
```

SOUND SUCCESS

Room For Manoeuvre
The company headquarters are in Sutton Park, outside Reading. Audiogenic moved here in April 1984 from a house in the centre of Reading that could no longer provide enough storage space

Audiogenic may not be as well known as other software companies, but it has built up a good reputation as a major supplier of software for Commodore machines. Originally merely a tape duplication company, Audiogenic now writes and distributes software and manufactures hardware peripherals.

Audiogenic is described by its founder and managing director Martin Maynard as 'a marketing and manufacturing business'. Maynard originally worked in the music industry, and established Audiogenic in Reading in the early 1970s as a recording studio and audio tape duplication service. In 1978, Audiogenic was commissioned by the Southern Electricity Board to duplicate a computer data cassette. The company's equipment was modified to cope with the demands of producing computer tapes in bulk, and Audiogenic signed a contract with Commodore to handle software duplication for the PET microcomputer.

The company then took over the marketing and distribution of Commodore's cassette catalogue, and began selling books, magazines and other Commodore-related equipment. After the launch of the Vic-20 in 1981, Maynard obtained licences to market Vic products developed by American software houses. Software manufactured under licence still accounts for 80 per cent of the Audiogenic catalogue, and represents 85-90 per cent of the company's £1.7 million turnover.

This emphasis on software marketing, rather than in-house production, appears to have been very successful, and Maynard estimates that Audiogenic has now produced over one million cassettes. 'Our strongest point is that we have a large catalogue. This diversity means that we can understand what's going on in the market,' Maynard claims. 'Having been in the software industry for six years, we've seen it all before. Of all the software sent out over the last year, 20-30 per cent is still on someone's shelf. Writers are losing touch with what people want.'

This cautious approach, promoting tried and tested software, contrasts strongly with the go-for-broke tactics employed by many other software houses. David Smithson, the Audiogenic product manager, points out: 'You don't see us buying Porsches. Some companies are just setting themselves up for a fall.'

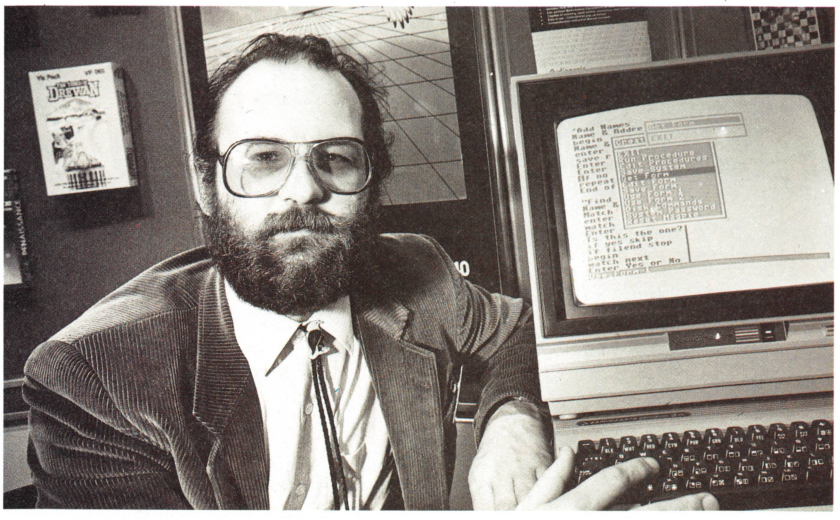
Audiogenic now employs 25 people. The company's leading programmer, Dave Middleton (writer of the highly rated Magpie database package), is employed on a freelance basis. Audiogenic tends to concentrate on utility software for its in-house programs, rather than going for the fast profits associated with the games market. As Maynard explains, 'Computers will always have a games element but that phase is now dying and computer software will develop into something more useful. When a package is selling two to three hundred a month you think "that's not selling very well", but it will still be selling that number in a year's time.'

This does not mean that the company neglects the games market entirely. One of Audiogenic's biggest sellers was Motor Mania, and the company has recently launched Alice in Videoland for the Commodore 64. Developed under licence, this game consists of a massive 90 Kbytes of code spread over five screens that are loaded from disk as the game progresses.

Shortly after moving into larger premises in March 1984, Audiogenic installed updated tape duplication equipment. The new machinery reproduces a program repeatedly on a continuous length of tape, which is cut up and packaged in cassette form after duplication. This is both faster and more convenient than the old system, which duplicated programs onto separate cassettes and meant that the company needed to store large stocks of blank C10 and C20 cassettes.

Audiogenic intends to continue its policy of distributing products manufactured by other companies in the UK and USA. Diversification into hardware peripherals has led to the company marketing a touch tablet graphics pad that was developed by Koala Technologies. Planned software projects include a range of cassettes for MSX machines and software for the new Commodore 16 home computer.

Leading Light
Audiogenic's Managing Director Martin Maynard who founded the company as a recording studio in the early 1970s



THE HOME COMPUTER ADVANCED COURSE

INDEX TO ISSUES 1 TO 12

THE HOME COMPUTER ADVANCED COURSE

INDEX TO ISSUES 1 TO 12